Simultaneous Localisation and Mapping with Prior Information

Martin P. Parsley

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University College London.

Department of Computer Science

University College London

2010

I, [Martin Parsley], confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signed: _____

Abstract

This thesis is concerned with Simultaneous Localisation and Mapping (SLAM), a technique by which a platform can estimate its trajectory with greater accuracy than odometry alone, especially when the trajectory incorporates loops. We discuss some of the shortcomings of the "classical" SLAM approach (in particular EKF-SLAM), which assumes that no information is known about the environment a priori. We argue that in general this assumption is needlessly stringent; for most environments, such as cities some prior information is known. We introduce an initial Bayesian probabilistic framework which considers the world as a hierarchy of structures, and maps (such as those produced by SLAM systems) as consisting of features derived from them. Common underlying structure between features in maps allows one to express and thus exploit geometric relations between them to improve their estimates. We apply the framework to EKF-SLAM for the case of a vehicle equipped with a range-bearing sensor operating in an urban environment, building up a metric map of point features, and using a prior map consisting of line segments representing building footprints. We develop a novel method called the Dual Representation, which allows us to use information from the prior map to not only improve the SLAM estimate, but also reduce the severity of errors associated with the EKF. Using the Dual Representation, we investigate the effect of varying the accuracy of the prior map for the case where the underlying structures and thus relations between the SLAM map and prior map are known. We then generalise to the more realistic case, where there is "clutter" - features in the environment that do not relate with the prior map. This involves forming a hypothesis for whether a pair of features in the SLAM state and prior map were derived from the same structure, and evaluating this based on a geometric likelihood model. Initially we try an incremental Multiple Hypothesis SLAM (MHSLAM) approach to resolve hypotheses, developing a novel method called the Common State Filter (CSF) to reduce the exponential growth in computational complexity inherent in this approach. This allows us to use information from the prior map immediately, thus reducing linearisation and EKF errors. However we find that MHSLAM is still too inefficient, even with the CSF, so we use a strategy that delays applying relations until we can infer whether they apply; we defer applying information from structure hypotheses until their probability of holding exceeds a threshold. Using this method we investigate the effect of varying degrees of "clutter" on the performance of SLAM.

Acknowledgements

Firstly I would like to thank my supervisor Simon Julier for his outstanding contributions in terms of time, dedication and enthusiasm during the PhD, and my secondary supervisor Bernard Buxton for his detailed comments and discussions. I would also like to thank my friends for their friendship and support, in particular Min H. Kim for his invaluable technical support during a critical time. I would also like to thank my parents for their patience and support.

Finally, I would like to thank my viva examiners Andrew Davison and Marek Ziebart, and my intermediate viva examiners Simon Prince, Danny Alexander and Paul Groves for their valuable input. I would also like to acknowledge and thank the SEAS-DTC partners at BAE systems, including Hector Figueiredo, Tim Doggart and Alex Stewart, who facilitated the collection of data in London.

Image acknowledgements:

Figure 2.5 - © Collins Bartholomew Ltd.

Figures 5.1 and 6.12 - © 2008 NAVTEQ/ © AND / © 2007 Intermap / © 2008 Blom

Figure 6.2 - © Crown Copyright/database right 2010. An Ordnance Survey/EDINA supplied service.

Other acknowledgements:

Parts of this thesis were partially supported by a UK Research Council EPSRC Doctoral Training Award number EP/P502802/1 and partially by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence (project AA018).

Contents

List of Figures

List of Tables

List of Acronyms

CSF	Common State Filter
DOF	Degree of Freedom
EKF	Extended Kalman Filter
EM	Expectation Maximisation
FLOP	Floating Point Operation
GIS	Geographical Information Systems
GPS	Global Positioning System
JCBB	Joint Compatibility Branch and Bound
LIDAR	Light Detection and Ranging
MAP	Maximum a Posteriori
MC	Monte Carlo
MHSLAM	Multiple Hypothesis SLAM
MHT	Multiple Hypothesis Tracking
MI	Mutual Information
ML	Maximum Likelihood
MSE	Mean Squared Error
NEES	Normalised Estimation Error Squared
OS	Ordnance Survey
rMSE	Root MSE
SEIF	Sparse Extended Information Filter
SICK	(A manufacturer of LIDAR sensors)
SLAM	Simultaneous Localisation and Mapping
UAV	Unmanned Aerial Vehicle
UCL	University College London
UGV	Unmanned Ground Vehicle
UKF	Unscented Kalman Filter

Notation

$\mathbf{x}$	True state
$\hat{\mathbf{x}}^h$	Estimated state (mean) of the hypothesis h
$\mathbf{x}_v$	Vehicle state
$\mathbf{x}_p$	State for map or feature type p
$\mathbf{P}^h$	State covariance of the hypothesis h
$\hat{\mathsf{x}}$	Common State mean (CSF)
P	Common State covariance (CSF)
$\hat{\mathsf{x}}^h$	Hypothesis state mean of hypothesis h (CSF)
P^{h}	Hypothesis state covariance of hypothesis h (CSF)
w^h	Weight of the hypothesis h (MHSLAM)
σ	Standard deviation
$p(i)$	Probability of i
$\aleph(\mu, \Sigma)$	Gaussian distributed noise with mean μ and covariance Σ
η	Generic normalising constant
$\mathbf{I}(a, b)$	Mutual Information between a and b
$O(.)$	Big O complexity notation
$\chi^2_c\|_{95\%}$	95% confidence bound of the χ^2 distribution with degrees of freedom c
$\mathbf{z}$	Observation state
$\mathbf{w}$	Observation noise
$\mathbf{R}$	Observation noise covariance
$\mathrm{h}(.)$	Observation model
$\nabla\mathbf{H}$	Observation model Jacobian
$\mathbf{M}$	Observation model Hessian
$\mathrm{g}(.)$	Prior map observation model (Dual Representation)
$\nabla\mathbf{g}$	Prior map observation model Jacobian (Dual Representation)
$\hbar(.)$	Inverse observation (initialisation) function
$\nabla\hbar$	Inverse observation function Jacobian

$\mathbf{u}$	Control input state
$\mathbf{v}$	Control noise
$\mathbf{\Sigma}$	Control noise covariance
$\nabla\mathbf{G}$	Control inputs Jacobian
$f(.)$	Process model (transition function)
$\nabla\mathbf{F}$	Process model Jacobian
$\mathbf{Q}$	Process noise covariance
$\mathbf{Q}_s$	Stabilising noise covariance
ν	Innovation
$\mathbf{S}$	Innovation covariance
$\mathbf{K}$	Kalman gain
$\mathbf{W}$	Indicator matrix for the Kalman gain
$\triangle t$	Time increment (difference between successive time steps)
k	Time step
K	Set of time steps, generally $1:k$
$i:j$	Set of time steps from i to j
$\mathbf{x}(i)$	State at time step i
$\hat{\mathbf{x}}(i\|j)$	Estimate at time step i given observations up to and including time step j
B	Vehicle (platform) wheelbase
u_v	Vehicle (platform) speed
$\mathbf{c}$	Data association parameter
G_p	Feature map model for the feature map p
H_p	Observation model for the feature map p
$\mathbf{s}_{w\{s\}}$	Instance s of structure class w
$d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_w\{s\}\})$	Indicator of feature map $\mathbf{x}_{p\{i\}}$ being a member of the structure $\mathbf{s}_{w\{s\}}$
$d_{in,s}$	Abbreviated form of $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_w(s)\})$
$\mathbf{f}\left(\mathbf{x}_p, \mathbf{x}_n\right)$	Relationship between feature maps p and n
$\nabla\mathbf{f}$	Jacobian of the relationship between feature maps, $\mathbf{f}(.)$
θ_{pn}	Parameter encoding a degree of freedom in the relationship $\mathbf{f}\left(\mathbf{x}_p, \mathbf{x}_n\right)$.

Chapter 1

Introduction

1.1 What is SLAM?

With the increasing use of autonomous agents (platforms) such as unmanned aerial vehicles (UAVs) and rescue robots in both military and civilian applications, there is a need for these agents to be able to *localise* themselves, that is determine their *pose* (location and orientation) in space. This is important for an autonomous agent, as most tasks it performs will require it to interact with its environment in some way. For such an agent to be truly autonomous it must maintain some notion of where it is, so that it can make decisions on where to travel to next or what actions to take at a particular location, or to inform its controllers of its location. There are a number of applications requiring such localisation, for instance the operation of dump trucks within an underground mine [79], exploring and mapping disaster sites while locating potential victims and threats [63, 91, 65], and in augmented reality (wearable computing) [27].

Simultaneous Localisation and Mapping (SLAM) is a technique whereby such an agent equipped with sensors (such as a camera) can create a spatial map of its environment, while simultaneously localising itself relative to this map. Figure 1.2 shows this process for a small robot equipped with a laser scanner. As the robot moves around the environment, static, repeatedly-observable features in the environment (such as building corners) are detected from the laser scanner measurements, and form the basis of an internal map; the robot is able to localise itself in terms of the features in this map. The benefit of SLAM is that when the robot re-observes features in its map, the uncertainty in its position will generally reduce to similar levels to those of the features, as the uncertainty in the position of a static feature remains bounded over time. The requirement for SLAM has come about partly due to the inadequacy of alternatives for localisation such as *odometry* (wheel and steer encoders) and global positioning systems (GPS).

While odometry is easy to deploy [45], wheel and steer encoders suffer from significant drift [22, 127, 45]. Figure 1.1 shows an example of the poor trajectory estimate obtained when solely using odometry; other sources experience similar errors [35, 121, 43, 79].

An alternative to using encoders for odometry is *visual odometry*, a method for estimating motion based on visual input from cameras mounted on the platform. Visual odometry can provide a velocity and position estimate that exhibits far less drift than wheel encoders; generally around 1% of distance

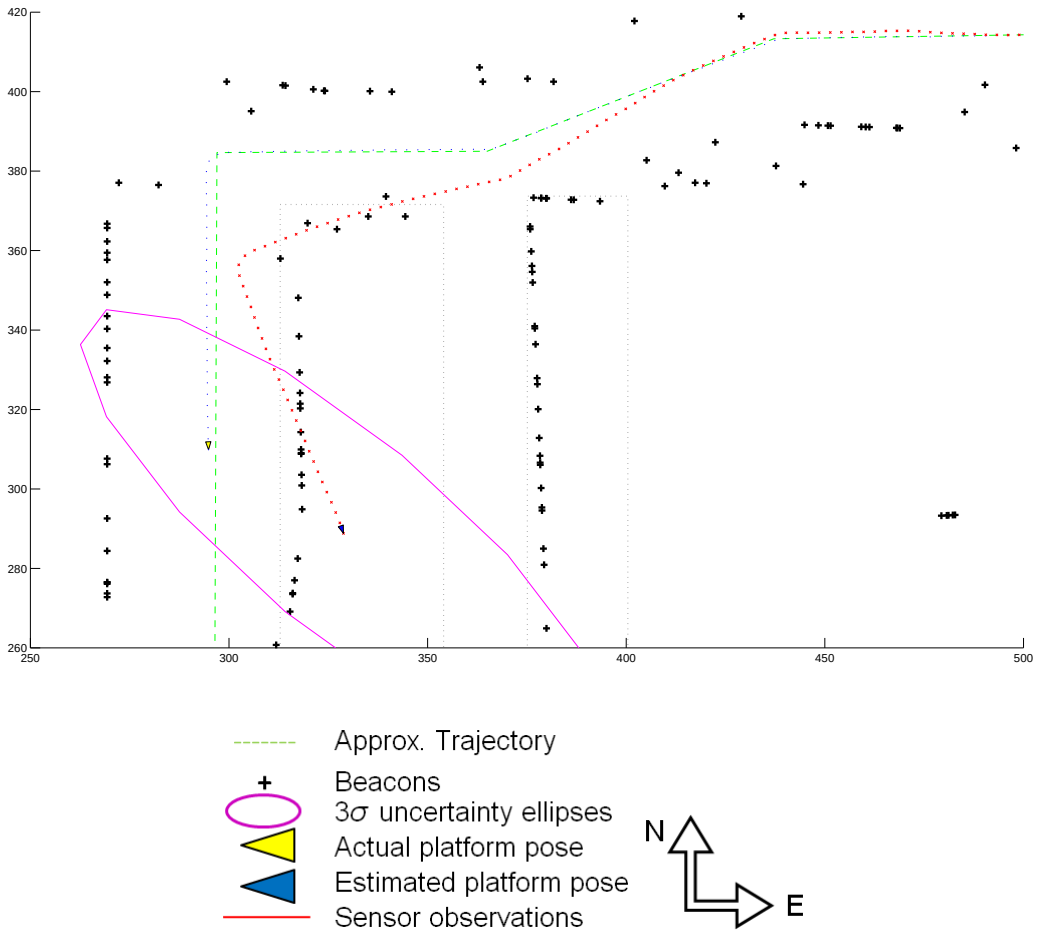

Figure 1.1: Example from our simulation environment (units in m) showing the trajectory estimate (small $\times$'s) for a simulated platform using odometry only (thus ignoring the beacons), compared to the actual trajectory (dots). Significant drift has built up leading to a large pose error. Over long distances the degree of error is unacceptably high. The large ellipse represents the 3σ bound of the pose estimate.

travelled, however as little as 0.1% is possible [67]. Although this drift is small, for prolonged operation over great distances (of the order of kilometres) the error will ultimately accumulate to a greater extent than if complemented with SLAM [90], particularly in trajectories involving loops (where the agent returns to an area it has previously mapped).

Alternatives to odometry include the use of a compass and GPS. A compass could be used to give an absolute estimate of a platforms orientation, however on its own it is inadequate as it is subject to significant error when used in places where the Earth's magnetic field is disrupted, for instance in buildings with a significant metal content [41], or on planets without a magnetic field. GPS can be used to give a location estimate, however it suffers from problems with reception around buildings ("urban canyons") [45] and in other environments, such as underground [49]. Naturally GPS cannot be used where there is no GPS infrastructure, such as on other planets.

SLAM can be used with the sole aim of providing localisation, in which case the map produced is a by-product and may only be of secondary importance. This is in contrast with photogrammetry, where ultimately the map is most important and accurate localisation of the platform arises as a means to this end. In addition, SLAM is more difficult as it is a continuous process that is performed in real time, whereas photogrammetric data is often processed offline.

An alternative to building up a map during SLAM from which to localise is to populate the environment a-priori with identifiable features (beacons) with known locations, which the agent can observe and use to localise itself. However the investment required to survey the environment and mount beacons is costly and restricts the application to environments that can be accessed a priori, making SLAM the more desirable method in many cases [79, 45]. Another example is that of RHINO [118, 120], a robot system first deployed in the Deutsches Museum Bonn as an interactive tour guide. This required very accurate maps (of the order of centimetres), which were acquired laboriously through manual surveying previously. The current system installed in the Carnegie Museum of Natural Science used the robot to generate the map, with a saving in time and effort.

Figure 1.2 shows an example of the kind of SLAM we are interested in; that of a robot performing SLAM in a large urban environment. The crosses represent the internal map estimate. At the same time, the platform knows its location in the environment based on this map. In general it is assumed that the platform has no prior knowledge of its environment, however in most cases its starting position is known. Sensors mounted to the platform, such as laser range finders or cameras detect suitable features (interchangeably called *landmarks* or *beacons*) in the environment, such as building edges or distinctively textured areas, and use these to build up a representation (i.e. map) of the environment in terms of these features. Simultaneously, this map is used to localise the platform. As noted above, in its pure ideal form, SLAM eliminates the need for artificial infrastructures, beacons or a priori topological knowledge of the environment [32].

There are a number of sensor types used to obtain measurements when performing SLAM [80]. Each method has its advantages and disadvantages, and can affect the requirements and performance of the SLAM problem [43, 101, 111]. In general there are two qualitatively distinct types of sensors;

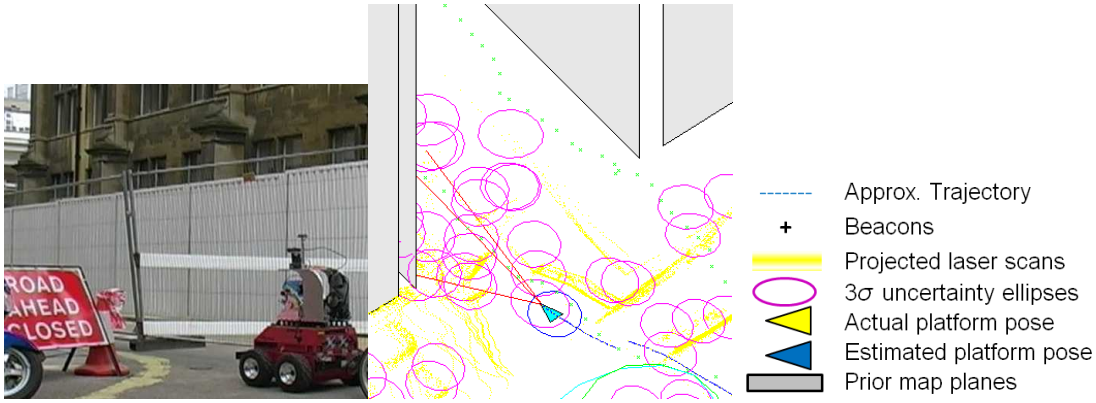

Figure 1.2: Example of SLAM showing the robot used in chapter 6 performing SLAM in an urban environment (left) [24], and a visualisation of the SLAM process showing the uncertain robot position, path and beacon estimates, and planes representing building walls (right). Here prominent features in the environment (such as building corners) are represented as point beacons in the SLAM state, along with the uncertainty in their position estimate.

range-bearing (such as LIDAR, including SICK scanners) and bearing-only (mainly cameras).[1]

1.2 Sensors Types used in SLAM

There is a significant difference between range-bearing sensors and cameras in SLAM. Range-bearing sensors give the range and bearing to the beacons, and thus allow for *range-bearing* SLAM. Cameras only give the bearing, and thus require a type of SLAM called *bearing-only* SLAM.

The most common range-bearing sensor type used in SLAM is LIDAR (light detection and ranging), where a laser beam is systematically scanned across the environment at high frequency, yielding a dense set of points with accurately known range and bearing from the sensor. The most common 2D laser scanner is the SICK scanner, which while scanning in a single plane, can be mounted and rotated so as to obtain high resolution point clouds of the environment in 3 dimensions [125].

There has been a recent surge of interest in bearing-only SLAM as it allows for the use of a single camera as a sensor rather than the LIDAR scanners used in range-bearing SLAM. A camera has a number of advantages compared to a LIDAR scanner. It can be applied as a cheap, compact sensor and is particularly suited to emerging areas such as wearable robotics, telepresence and television [25, 27, 70]. In addition, it is a passive sensor and thus safe to operate, and will not interfere with any devices in its environment. This is particularly desirable for military applications and for performing SLAM in an urban environment. A camera may also leverage a growing number of computer vision techniques when performing data association, and may recognise certain features of its environment [74, 119, 107, 73]. An example is the use of image data to detect when loop closing has occurred [24].

Davison [25, 27], Calway [44] and Klein [64] are researching SLAM using a single wearable cam-

[1] We will not consider range-only sensors here, suffice to say that they include sonar [80], and are mainly used underwater. However the principles developed in this thesis can still be applied to SLAM with these sensors.

era, for use in wearable sensors, game interfaces and other devices.

1.3 Motivation for prior information in SLAM

Despite the popularity and potential of SLAM, there is difficulty in applying it to real-world applications. Figure 1.3 shows an example of a map produced using a popular method based on the Extended Kalman Filter [23]. The points represent features in the map, the positions of which the SLAM algorithm is estimating. The ellipses around the features shows the 3σ bound for where we expect each feature to lie. Note how the map estimate appears rotated relative to the actual positions of the features, which lie outside of their ellipses (and thus the map does not reflect where the actual features are). For a map that is being estimated correctly it should be unlikely for the features not to lie within the ellipses, however this occurs relatively often, and thus the filter is termed inconsistent.

The primary cause of the inconsistency in this example is that the EKF is unable to correctly model the angular uncertainty in the system [58, 61]. Error also arises from the linearisation of the process and observation models. Inconsistency in SLAM can also arise from unmodelled errors, such as wheel slip, or incorrectly associating observations with beacons in the map (known as the *correspondence* or *data association* problem). The nature of SLAM is such that any errors made will propagate over time, affecting all subsequent estimates.

There is a storage and computational penalty associated with estimating the map and platform pose in SLAM. For standard EKF-SLAM this increases nonlinearly with the number of features in the map, meaning that real-time performance cannot be maintained indefinitely.

Most approaches to SLAM consider the problem from a perspective in which the platform moves through an environment about which it has effectively no prior knowledge [22, 32, 28]. Thus, the only map that the platform has for localisation is the corrupted map created by its own noisy sensors based on its noisy motion. Research to date has concentrated on reducing the degree of corruption in the map, and improving the computational tractability of SLAM, however the reliance on a single platform for mapping means that it is very difficult to apply SLAM robustly over prolonged time periods in real environments.

However in many cases a number of different agents will have mapped the same environment. For example, a military UAV may be performing SLAM in an area that has already been mapped by a satellite. Thus an agent performing SLAM is rarely in completely unmapped territory. In addition, environments rarely consist of an alien landscape with unrecognisable features. These prior maps and semantic knowledge of the environment constitute prior information that can contribute vastly to localisation and mapping. Yet most current SLAM methods are unable to exploit these sources of information, due to their unduly restrictive assumption that the environment is completely unknown in advance [119]. Given the difficulty of the SLAM problem, particularly in practical, outdoor environments, a number of researchers are looking to improve SLAM by exploiting information known about the environment, rather than solely attempting to improve the algorithms [20, 12, 102, 69].

In this thesis we will show that using prior information in SLAM has the potential to improve the accuracy, robustness and performance of SLAM algorithms . Figure 1.4 shows an example of the

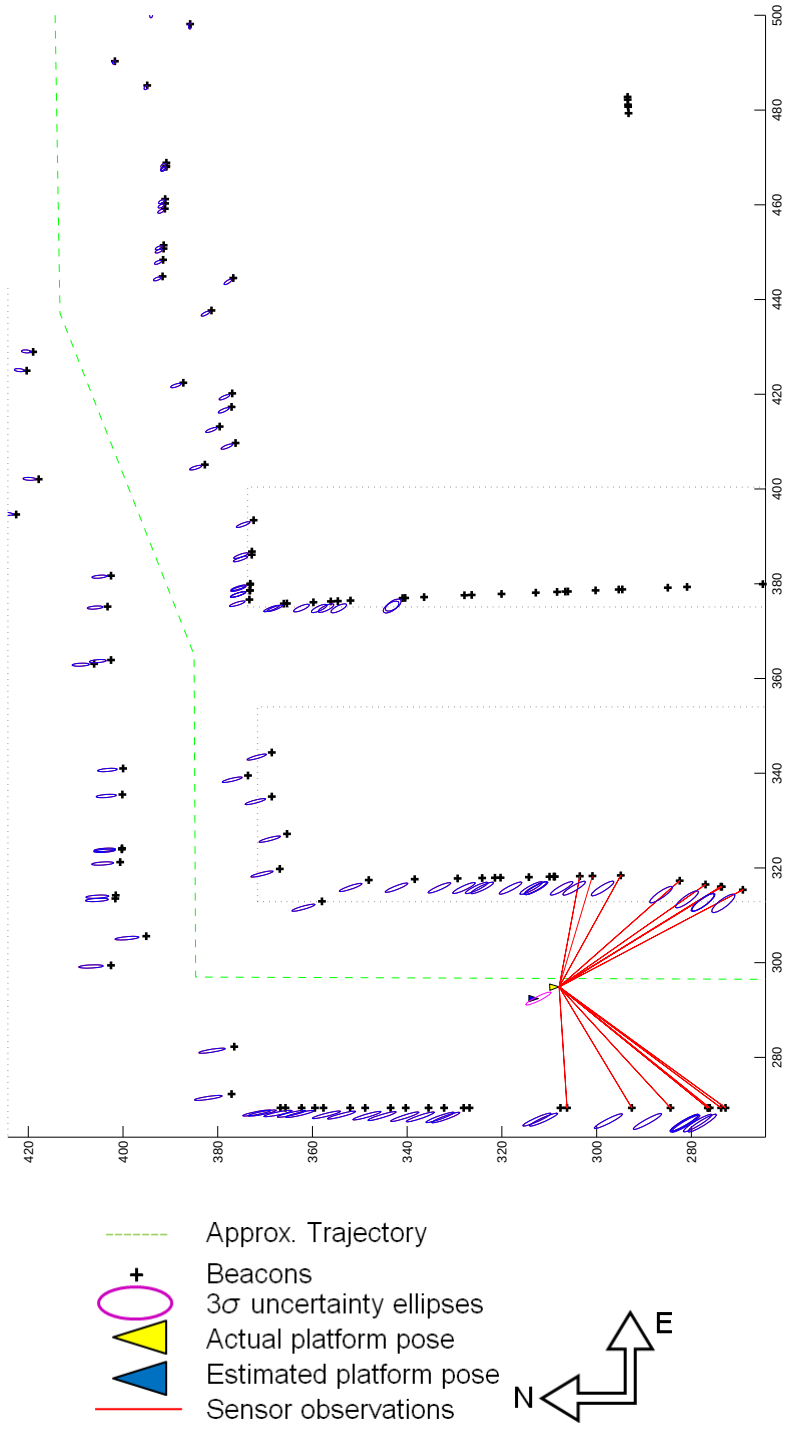

Figure 1.3: Example of a map and estimates produced by a SLAM system, obtained from one of our simulation runs (units in m). The crosses represent the actual positions of beacons in the environment, while the ellipses represent 3σ bounds on the estimate of their position. The SLAM system here is deemed inconsistent because the actual beacons do not lie within their position estimates, suggesting the degree of error is greater than its estimate.

improvement that is possible compared to Figure 1.3, when a noisy geometric prior map consisting of line segments (with a 1 m standard deviation error compared to the 0.1 m range error of the sensor) is introduced into the SLAM process. With the exception of the prior map, both SLAM runs are the same, using identical observations and noises.

1.4 Scope

The SLAM problem is very active with research ongoing in a number of areas. Given the number of unsolved issues that are part of the complete SLAM problem, we restrict ourselves to focusing on a particular SLAM problem. Thrun [114] offers a convenient taxonomy of localisation problems in the following taxons:

- Localisation: This concerns the type of knowledge available to the robot at the start of and during the run. There are three types of localisation problems, each progressively harder. *Position tracking* assumes we start with a known pose (with a degree of uncertainty). *Global localisation* is more difficult, and assumes that the robot does not know its initial pose. Finally the *kidnapped robot* problem is one where the robot (platform) is suddenly teleported to another location on the map without its knowledge. This primarily tests the ability of the robot to recover from localisation failures.

  In this research we restrict ourselves to the simpler position tracking (with a known uncertain pose), with the aim that our research could be extended to global localisation and kidnapped robot cases.

- Environments: These are either *static* or *dynamic*. A static environment is one in which only the platform moves, and all other objects remain fixed indefinitely. Dynamic environments are harder but more realistic, and may consider both short term motion, for instance walking pedestrians and longer term motion such as temporarily parked vehicles.

  Because dynamic environments are an entire research field in themselves [14], we only consider static environments in this research. While we do perform SLAM in a dynamic environment in chapter 6, we treat the dynamic components as noise.

- Controllability: A robot may be *actively* controlled by the localisation algorithm, for instance in autonomous exploration [127]. Alternatively, in *passive* localisation the robot may be controlled by other means, with the localisation algorithm only able to observe the robot motion.

  We consider passive localisation in this thesis. By not considering the robot exploration strategy, our research is applicable to both active and passive localisation. Active localisation requires cost functions, which themselves are modelled on the environment. By using prior information, the platform's knowledge of its environment is improved, allowing for more detailed and accurate cost functions.

- Number of robots: SLAM may be performed simultaneously by a single robot, or by a group of

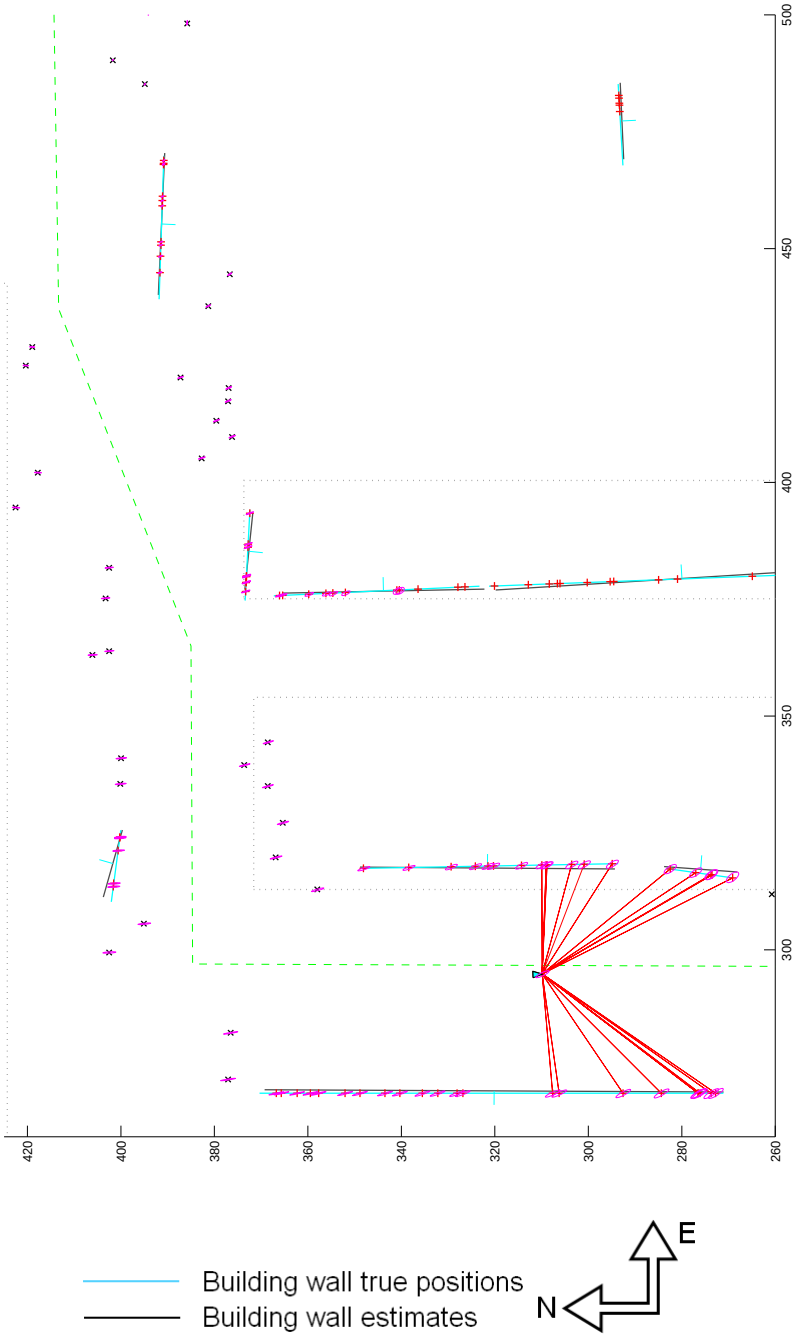

Figure 1.4: Simulated SLAM run from Figure 1.3 (units in m), with the introduction of a noisy prior map consisting of line segments with a 1 m standard deviation error to the SLAM process. This results in an accuracy and consistency improvement compared to SLAM without the prior map (contrast with Figure 1.3). This is true despite the range-bearing sensor on the platform being an order of magnitude more accurate than the prior map, having a 0.1 m standard deviation range error.

robots. Multiple robot SLAM raises issues of communication and data fusion, which are research topics themselves and thus outside the scope of this thesis.

Thus to avoid having to consider multiple platforms, we will consider prior information in the context of platforms performing SLAM serially, that is a robot performing SLAM using information from a platform that previously performed SLAM in the same area (but not at the same time). Thus though the prior information the platform uses may be incorrect or inaccurate, it is accessible to the platform when SLAM starts, and is not subject to change thereon.

1.5 Aims

We aim to perform a comprehensive study on the integration of absolute prior information into SLAM, featuring:

- A rigorous method by which a platform performing SLAM may make immediate use of absolute prior information, with initial emphasis on prior geometric maps.
- An initial analysis of some of the issues surrounding its use in SLAM, such as the effect of the quality of the information, its sparsity and correctness.
- Demonstration of a system performing SLAM utilising a prior map in a large practical environment.

In developing a method for applying prior map information in SLAM, there are a number of issues and factors we will consider. We will begin to address whether introducing prior constraints fundamentally alters the characteristics of SLAM, and the extent to which they can potentially improve SLAM. Specifically, we consider:

1. Quality: What effect does the prior map accuracy have on SLAM accuracy and consistency? Ultimately, a question that would form part of the broader research in this area is whether any prior map, regardless of how accurate is worth exploiting, or whether there is a point at which the prior map makes no difference to the SLAM problem, and is thus not worth using.

   We have considered the effect of the prior map when the relationships between the features in the prior map and SLAM state are known (chapter 3) and unknown (chapters 4 and 5). Even a prior map with a 1.5 m 1σ (1 standard deviation) error can improve the accuracy of the estimate and reduce the degree of inconsistency in EKF-SLAM, however these benefits decrease substantially as the prior map error increases. At a prior map accuracy level of 1 m, the improvement in consistency and accuracy over regular SLAM is very noticeable, as shown in Figure 1.4.

2. Sparsity: How much prior data would have to be applied, and how often? Can very sparse maps or conservative use of prior information still improve the SLAM map?

   We used a relatively sparse prior map for our experiments, consisting of 48 line segments over a 1.4 km trajectory. We have found that even this level of sparsity significantly improves the

accuracy and consistency of the SLAM estimate. However at high levels of ambiguity, which we term *clutter* (considered in chapter 5) the ability to resolve the relationships between the SLAM state and prior map decreases to the point where very little of the prior map information can be used.

3. Correctness: The prior map may contain qualitative errors such as spurious features. For instance, a feature in the prior map may no longer exist in the real world, as in the case of an outdated map. We would like to know whether we can account for such errors and filter them out of the SLAM process. A possible application where this could be used deliberately is in model validation; to use SLAM to determine whether the environment, for instance a new building, significantly deviates from the blueprints (prior map).

   We propose a theoretical probabilistic framework in chapter 3 which would allow the system modeller to account for these problems, and show that our EKF-based implementation maintains a low association error rate when 6 out of 54 line segments in the prior map are spurious.

4. Integration: In general the way features are parameterised in the prior map will be different to those in the SLAM state (they will be *heterogeneous*), thus can this information be easily integrated into the state?

   We considered a geometric prior map consisting of line segments, used to inform point beacons in the SLAM state. In chapter 3 we found that integrating this map into the SLAM state in a robust way is non-trivial, due to nonlinearity and EKF errors. These can nullify the improvement that using information from the prior map gives, and in our case necessitated the use of the Second Order filter, along with the development of the Dual Representation, a parameterisation specifically designed to prevent corruption of the prior map.

5. Robustness: How well does the application of prior information work in practice?

   With a parameterisation designed to use the prior map to mitigate EKF errors, in chapter 5 we found that the severity of the errors in EKF-SLAM are such that we can be optimistic when resolving ambiguity between features in the SLAM state and prior map, using more of the prior information to offset EKF errors, even if our false positive rate is higher (applying the prior information to beacons to which it does not apply). When the prior map is very accurate, for instance 10 cm 1 std. dev. error the system is more susceptible to false positives stemming from EKF errors, however less accurate prior maps, such as 1 m std. dev. are more robust to this.

6. Applicability: Ideally prior information would be applicable to all SLAM situations, i.e. all SLAM methods, platforms and sensors. However this is unlikely to be the case. There may be situations in which the use of prior information may not be appropriate, or would be particularly challenging. Conversely, if we find that there are situations under which SLAM with a prior map works particularly well, we will seek to identify these.

1.6 Structure of the Thesis and Contributions

The structure and contributions of this thesis are as follows:

- In chapter 2 we provide the problem statement and illustrate the characteristics that make SLAM difficult, with emphasis on EKF-SLAM. We illustrate some alternative approaches, however all have significant disadvantages. This motivates the need to consider prior information.

- chapter 3 presents a novel probabilistic framework for reasoning with prior information. The framework allows one to express models of the shared information between heterogeneous prior maps based on common underlying structure in the environment from which features in the maps were generated.

  We also present the novel *Dual Representation*, a method for using a geometric prior map incrementally in EKF-SLAM such that EKF errors are mitigated. We apply this method to investigate the effect of the accuracy of the prior map on platform localisation, when correspondences (caused by the common underlying structure) between features in the SLAM state and prior map are known.

  However, known correspondences are an ideal situation; in practice they are ambiguous and would have to be inferred by the SLAM system.

- We address this problem in chapter 4 by showing how an incremental multiple hypothesis SLAM (MHSLAM) approach can be used to estimate the correspondences. We present the *Common State Filter* (CSF), a novel method for efficiently resolving ambiguity in sparse EKF-based MHSLAM. We apply this method with the Dual Representation to incrementally resolve the ambiguous correspondences between the prior map and the map being built by SLAM.

  However MHSLAM can be computationally expensive as the number of hypotheses still grows exponentially, even with the CSF. This motivates a more computationally tractable delayed approach.

- In chapter 5 we show how the correspondences can be estimated using a more computationally efficient delayed approach. This also allows inference of the underlying structure to be informed by a prior. We compare this approach with CSF-based MHSLAM, and use it with the Dual Representation to investigate the effect spurious features in the prior map, its accuracy and the level of clutter (proportion of beacons that correspond with the prior map) on SLAM.

- In chapter 6 we apply the method in chapter 5 to perform one of the first practical experiments on applying a geometric prior map in SLAM in a real environment, using a vehicle equipped with a range-bearing laser scanner, from which we extract sparse salient points. The prior map consists of line segments representing the footprints of building walls.

- In chapter 7 we summarise the work done in this thesis, and our conclusions. We then discuss further work that could follow on from this research.

Chapter 2

Mapping and Localisation in an Uncertain World

In this chapter we give an overview of SLAM, presenting its general form and the main requirements that a SLAM algorithm should strive to meet, and discuss some of the problems that make it difficult to solve. We then present some philosophies to solving it, and a broad overview of techniques, with emphasis on the Extended Kalman Filter (EKF), and its shortcomings. In addition to the EKF, we briefly consider three commonly used SLAM methods; Sparse Extended Information Filters (SEIF), FastSLAM and Maximum Likelihood estimation (ML).

We start by briefly giving an overview of the form of metric SLAM.

2.1 SLAM Form

The general structure of metric SLAM, shown in Figure 2.1, is as follows. At time k, the true state $\mathbf{x}(k)$ consists of the platform pose $\mathbf{x}_v(k)$ and a set of n static beacons $\mathbf{x}_{1\ldots n}$,

$$\mathbf{x}(k) = \left[\begin{array}{cccc} \mathbf{x}_v^T & \mathbf{x}_1^T & \ldots & \mathbf{x}_n^T \end{array} \right]_k^T . \tag{2.1}$$

In the 2D case, the platform pose consists of its x, y position and the orientation θ,

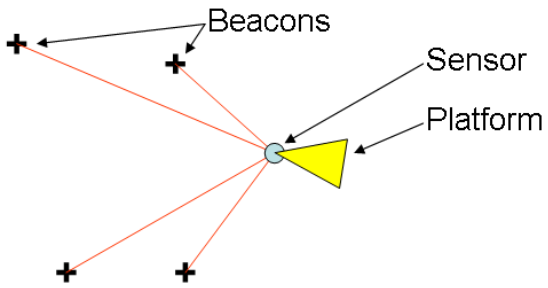

Figure 2.1: A metric SLAM scenario. The crosses represent point features in the environment that are repeatedly observed (represented by the rays) by a sensor mounted on a moving platform, represented here by a triangle.

$$\mathbf{x}_v(k) = \left[\begin{array}{ccc} x & y & \theta \end{array} \right]_v^T, \tag{2.2}$$

and each beacons state consists of the beacon x and y position,

$$\mathbf{x}_n = \left[\begin{array}{cc} x_n & y_n \end{array} \right]^T. \tag{2.3}$$

The SLAM estimator maintains an estimate of this state, with a notion of the uncertainty of its estimate. When the platform receives sensor information, it predicts its state estimate to the time at which the sensor information was received, and then uses it to update (correct) its estimate. This is shown as a flowchart in Figure 2.2. The platform is initialised (placed in a location in the environment) and the map is empty. The platform pose forms the reference for the map that will be created. SLAM is then an iterative process whereby at each time step, the platform *predicts* where it has moved to, and receives beacon observations from its sensors. By associating beacons from the sensors with beacons in the map, the platform can add new beacons to the map or *update* it based on observations of beacons already in the map.

By assumption, the beacons are stationary. Therefore, only the platform pose $\mathbf{x}_v$ has a motion model, which at time k is given by

$$\mathbf{x}_v(k) = F[\mathbf{x}_v(k-1), \mathbf{u}(k), k, \mathbf{v}(k)], \tag{2.4}$$

where $\mathbf{u}(k)$ are the control inputs with control noise $\mathbf{v}(k)$. For the case where the platform is a vehicle with a steered bicycle motion model [61], as used in this thesis, the control inputs are velocity u_v and steer u_s

$$\mathbf{u}(k) = \left[\begin{array}{c} u_v \\ u_s \end{array} \right]_k. \tag{2.5}$$

An observation $\mathbf{z}(k)$ can arise from one of three sources – either it is an observation of a beacon already in the map, a new beacon not present, or clutter (observations that do not correspond to features of interest). When the status of the observation is known then implementing SLAM is, in principle, very simple. The observation function that links the observation $\mathbf{z}(k)$ of the jth beacon $\mathbf{x}_j$ to the platform pose $\mathbf{x}_v(k)$ is

$$\mathbf{z}_j(k) \quad = \quad \mathbf{h}_j[\mathbf{x}_v(k), \mathbf{x}_j, \mathbf{w}(k)], \tag{2.6}$$

where $\mathbf{w}(k)$ is observation noise.

As mentioned in the previous chapter, SLAM has a number of uses, including facilitating navigation and exploration for autonomous robots. To be useful for these functions, there are a number of key requirements that a SLAM algorithm should meet. We will give a key overview of these below.

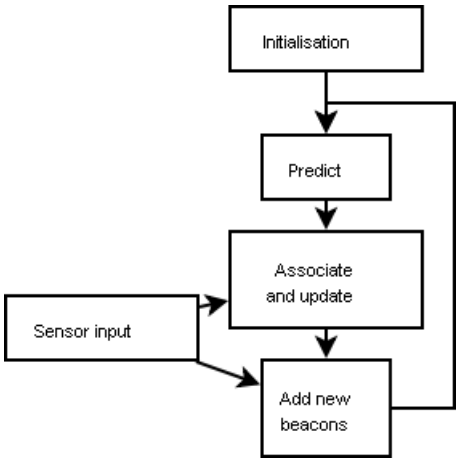

Figure 2.2: Flowchart showing the predictor-corrector structure of SLAM estimators.

2.2 Requirements for a SLAM Algorithm

In order for SLAM to be a feasible technique for a mobile platform to use, Frese [43] identifies three key criteria of importance in the design of a general SLAM algorithm, in terms of map quality, storage space and computation time [43, 105]. Because SLAM is applicable within a broad domain, the exact requirements that a SLAM algorithm needs to meet within each of these categories will vary depending on the details of the specific application and platform, however the general characteristics that SLAM algorithms should strive for as described by [43] are:

1. Optimality: the uncertainty of any aspect of the map should not be much larger than could be derived from the ideal estimator. That is to say, given the information available to the algorithm, it should not be unduly conservative or suboptimal. Ideally, under ideal conditions (for instance when the SLAM system models are representative of the actual system), certain optimality characteristics of the SLAM algorithm would be mathematically guaranteed.

   This is equivalent to stating that the map should represent nearly all the information contained in the observations, and implies the ability to represent the relative positions of beacons well, giving topologically consistent maps even when closing large loops. In this thesis we will use a SLAM method based on a mathematical technique for which there are proven optimality results; where extensions and assumptions are made that prevent these results being directly applicable, these are known and made not for convenience, but because mathematically rigorous techniques do not yet exist.

2. Consistency. A SLAM algorithm should not underestimate its uncertainty. An algorithm that does so is deemed to be *inconsistent*. The exact way in which this is defined depends on the algorithm used; a precise definition for the one used in this thesis will be given in section 2.4. Inconsistency is undesirable, as the degree of error in the estimate cannot be accurately gauged, and the ability of the algorithm to perform data association is reduced [121]. Again it would be desirable for this to be mathematically proven under ideal conditions.

3. Complexity: A SLAM algorithm needs to work incrementally (in real time). This is because an autonomous agent performs SLAM out of a need to have up to date knowledge of its location and surroundings. This places constraints on the storage and computational complexity of the SLAM algorithm. For the general case where computational resources are finite and the size of the environment is unbounded, the storage space of a map and incorporating sensor observations should be at worst linear in the number of beacons, otherwise a point will be reached when the computational requirements make incremental operation impossible [25].

   In this thesis we will not attempt to meet the linear storage and computational criterion directly, however instead we will identify and comment on the potential of prior information to aid in the meeting of this criterion in the future. In addition, there are a number of techniques for addressing computation in SLAM; we aim for our methods to be compatible with them.

There are a number of approaches to solving SLAM, however as we will discuss below, none of them are able to satisfy all three of these criteria in all situations. These approaches can be broadly segregated into three main types; we will now give a brief overview of these.

2.3 Philosophies to solving SLAM

The variety and difficulty of the problems associated with SLAM has led to several philosophies for solving it. There are three broad approaches; stochastic (estimation-theoretic), qualitative and numerical [32].

- Estimation-theoretic

  This is a stochastic approach and follows the framework established by Smith, Self and Cheeseman [108]. It is based on Bayes rule where the uncertainties over the vehicle and beacon positions are maintained. This method, which includes the EKF is the most popular and was briefly presented in Section 2.1. Durrant-Whyte [36] stresses the importance of explicitly representing errors and having an understanding of how they affect the geometry.

- Qualitative

  The qualitative approach to SLAM contrasts with the quantitative stochastic approach. It does not consider position and observations precisely. The main advantage to this approach is that accurate models of the system are not required, and it is less expensive computationally [32]. Kuipers and Byun [68] describe a qualitative method for robot exploration and mapping, with the aim of overcoming the fragility of purely metrical methods.

- Numerical

  The numerical approach does not use the rigorous stochastic formulation, but still retains a numerical approach to the problem [32]. Examples include Genetic Algorithm SLAM (GA-SLAM) [34] and Expectation Maximisation (EM) [119].

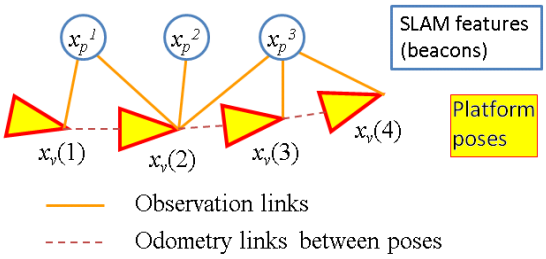

Figure 2.3: Graphical model of the full SLAM process for four time steps, showing the platform pose $\mathbf{x}_v$ and beacon $\mathbf{x}_p$ vertices; these are linked by observation $\mathbf{z}$ and odometry $\mathbf{u}$ edges. Observing the same beacon from multiple platform poses allows constraints to be formulated between the platform poses, improving their estimates and those of the beacons. In this way SLAM results in a better estimate than using pure odometry.

One of the most popular approaches to SLAM is the stochastic formulation, which models SLAM probabilistically and maintains probabilistic notions of the uncertainty in the SLAM system, and how it propagates. Figure 2.3 shows a graphical model for the full SLAM process. The platform poses $\mathbf{x}_v(k)$ at each time step k are linked together by odometry $\mathbf{u}(k)$ and the observations $\mathbf{z}(k)$ of a set of common beacons $\mathbf{x}_p$ (a subset of the SLAM map). For the time period $K = 1:k$, the posterior estimate in stochastic full SLAM is $p(\mathbf{x}_v(K), \mathbf{x}_p|\mathbf{z}(K), \mathbf{u}(K))$. In the *filtering* (as opposed to *smoothing*) approach to SLAM, the platform estimates of $\mathbf{x}_v(1:k-1)$ are marginalised out, and only the current platform pose estimate corresponding to $\mathbf{x}_v(k)$ is kept. The posterior at time step k, $p(\mathbf{x}(k)|\mathbf{z}(K), \mathbf{u}(K))$ can be updated from the posterior of the previous time step $p(\mathbf{x}(k-1)|\mathbf{z}(1:k-1), \mathbf{u}(1:k-1)$ in a recursive manner according to

$$p(\mathbf{x}(k)|\mathbf{z}(1:k-1), \mathbf{u}(K)) = \int p(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{u}(k))\, p(\mathbf{x}(k-1)|\mathbf{z}(1{:}k-1), \mathbf{u}(1{:}k-1))\, \mathrm{d}\,\mathbf{x}(k-1) \quad (2.7)$$

$$p(\mathbf{x}(k)|\mathbf{z}(K), \mathbf{u}(K)) = \frac{p(\mathbf{z}(k)|\mathbf{x}(k))\, p(\mathbf{x}(k)|\mathbf{z}(1:k-1), \mathbf{u}(K))}{p(\mathbf{z}(k)|\mathbf{z}(1:k-1), \mathbf{u}(K))}, \quad (2.8)$$

where $K = 1:k$, $\mathbf{x}(k)$ represents the joint vehicle and map state estimate at time k, $\mathbf{z}(K)$ all observations up to time k, $p(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{u}(k))$ is the platform transition model (the beacons remain stationary) and $p(\mathbf{z}(k)|\mathbf{x}(k)) = p(\mathbf{z}(k)|\mathbf{x}(k), \mathbf{z}(1:k-1), \mathbf{u}(K))$ is the observation likelihood. The Bayesian formulation is difficult to implement for arbitrary probability distributions, however propagating approximations to the first two moments of a distribution (the mean and covariance) is feasible provided the higher order terms are small, and is the approach taken by the Extended Kalman filter (EKF).

Although we will show that all current SLAM solutions suffer from problems that prevent them from meeting the three requirements discussed above, we adopt the EKF as our SLAM algorithm in this thesis. We will begin by giving a more in-depth description of its application to SLAM, and will then demonstrate the main problems that prevent EKF-SLAM from being an effective solution to the SLAM problem.

2.4 The Extended Kalman Filter (EKF)

The EKF-based approach is one of the most popular methods for performing SLAM in use currently [23, 45, 1, 89, 114, 79]. The EKF is an extension of the Kalman filter to non-linear systems, and has a recursive predictor-corrector structure, shown in Figure 2.2, allowing for incremental operation.

The EKF has the advantage of being well-studied and applied in many fields in addition to SLAM [114], such as nonlinear estimation and machine learning [124]. Its primary advantage is that it maintains a full covariance matrix, tracking the correlations between the estimated states. This is mandatory to prevent an erroneous map and inconsistent vehicle estimate [18, 32], and allows the EKF to update its entire map estimate consistently following loop closure [43].

At the heart of the system, the EKF represents the state distribution by its mean and covariance[1], which is propagated through the first-order linearisation of the nonlinear system [124, 101].

The EKF estimate of the state (2.1) consists of the mean and covariance

$$\hat{\mathbf{x}}(i|j) = \begin{bmatrix} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_1^T & \dots & \hat{\mathbf{x}}_n^T \end{bmatrix}_{i|j}^T = \begin{bmatrix} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_b^T \end{bmatrix}_{i|j}^T, \tag{2.9}$$

$$\mathbf{P}(i|j) = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{v1}^T & \dots & \mathbf{P}_{vn}^T \\ \mathbf{P}_{v1} & \mathbf{P}_{11} & \dots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{P}_{1n}^T \\ \mathbf{P}_{vn} & \mathbf{P}_{1n} & \dots & \mathbf{P}_{nn} \end{bmatrix}_{i|j} = \begin{bmatrix} \mathbf{P}_v & \mathbf{p}_{vb}^T \\ \mathbf{p}_{vb} & \mathbf{P}_b \end{bmatrix}_{i|j}, \tag{2.10}$$

where $(i|j)$ denotes the estimate at time step i given observations up to and including time step j at discrete time steps [75]. $\mathbf{P}_v$ is the vehicle pose covariance, $\mathbf{P}_{nn}$ is the nth beacon covariance, $\mathbf{P}_{vj}$ is the cross correlation between the vehicle and the jth beacon, and $\mathbf{P}_{nj}$ are the cross correlations between the nth and jth beacons. This joint covariance matrix must be maintained to ensure a *consistent* solution [61, 6, 108], as it captures the correlation between all the beacons and vehicle (i.e. the uncertainty structure of the map). For a filter to be consistent, its covariance should be greater than or equal to the true MSE of the estimate,

$$\mathbf{P}(k|k) \geq (\hat{\mathbf{x}}(k|k) - \mathbf{x}(k))(\hat{\mathbf{x}}(k|k) - \mathbf{x}(k))^T. \tag{2.11}$$

If the covariance is significantly larger, the filter is deemed conservative. A conservative filter is preferred to an inconsistent one, however neither is desirable; a suboptimal filter implies that it is not using all the information available, and thus should be improved.

The vehicle pose propagates according to (2.4), the noise requirements for $\mathbf{v}(k)$ being zero-mean with known covariance $\mathbf{\Sigma}(k)$. The conditional mean of the estimated vehicle pose $\hat{\mathbf{x}}_v$ propagates according to

$$\hat{\mathbf{x}}_v(k|k-1) = \mathcal{F}[\hat{\mathbf{x}}_v(k-1|k-1), \mathbf{u}(k), k, \mathbf{0}], \tag{2.12}$$

[1] Although it is commonly stated to be a Gaussian random variable (GRV), the EKF does not require the distribution to be Gaussian [10]. However, a Gaussian assumption is made in hypothesis testing later.

and the covariances according to

$$\mathbf{P}(k|k-1) = \nabla\mathbf{F}(k)\,\mathbf{P}(k-1|k-1)\,\nabla\mathbf{F}(k)^T + \mathbf{Q}(k), \tag{2.13}$$

where $\nabla\mathbf{F}(k)$ is the Jacobian of $F(\cdot)$ and $\mathbf{Q}(k)$ is the process noise covariance,

$$\mathbf{Q}(k) = \nabla\mathbf{G}\,\mathbf{\Sigma}(k)\,\nabla\mathbf{G}^T + \mathbf{Q}_s, \tag{2.14}$$

where $\nabla\mathbf{G}$ is the control inputs Jacobian, $\mathbf{\Sigma}(k)$ is the control noise covariance and $\mathbf{Q}_s$ is optional stabilising noise. Stabilising noise is a positive semi definite matrix sometimes added to improve the properties of the SLAM algorithm, generally to compensate for nonlinearities or modelling errors in the control inputs [57].

The control inputs Jacobian $\nabla\mathbf{G}$ takes the form

$$\nabla\mathbf{G} = \begin{bmatrix} \nabla\mathbf{G}_v & 0 & \dots & 0 \end{bmatrix}, \tag{2.15}$$

the v subscript referring to the vehicle, and the zeros to the beacons, as they are assumed static.

The observation function is given by (2.6), where the observation noise $\mathbf{w}(k)$ is zero-mean with known covariance $\mathbf{R}(k)$. Typically for a range-bearing sensor

$$\mathbf{R}(k) = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}_k, \tag{2.16}$$

where σ_r and σ_ϕ are the standard deviations of the range and bearing noise respectively.

If the beacon already has an estimate in the state, it is updated by linearising the observation function about the state estimate $\hat{\mathbf{x}}(k|k-1)$, then performing a Kalman update. The Jacobian $\nabla\mathbf{H}(k)$ for the observation function is *sparse* and has the form

$$\nabla\mathbf{H}(k) = \begin{bmatrix} \frac{d\mathrm{h}}{d\hat{\mathbf{x}}_v} & 0 & \dots & \frac{d\mathrm{h}}{d\hat{\mathbf{x}}_j} & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{d\mathrm{h}}{d\hat{\mathbf{x}}_v} & 0 & \dots & 0 & \frac{d\mathrm{h}}{d\hat{\mathbf{x}}_n} & 0 \end{bmatrix}, \tag{2.17}$$

where $\frac{d\mathrm{h}}{d\hat{\mathbf{x}}_v}$ are the matrices of appropriate dimension for the vehicle and $\frac{d\mathrm{h}}{d\hat{\mathbf{x}}_j}$ are the entries for the jth beacon.

The update is computed using the Kalman filter update equations,

$$\begin{aligned}
\nu(k) &= \mathbf{z}(k) - \hat{\mathbf{z}}(k|k-1) && (2.18)\\
\mathbf{S}(k) &= \nabla\mathbf{H}(k)\,\mathbf{P}(k|k-1)\nabla\mathbf{H}^T(k) + \mathbf{R}(k) && (2.19)\\
\mathbf{K}(k) &= \mathbf{P}(k|k-1)\nabla\mathbf{H}^T(k)\,\mathbf{S}(k)^{-1} && (2.20)\\
\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)\nu(k) && (2.21)\\
\mathbf{J}(k) &= \mathbf{I} - \mathbf{K}(k)\,\nabla\mathbf{H}(k) \\
\mathbf{P}(k|k) &= \mathbf{J}(k)\,\mathbf{P}(k|k-1)\,\mathbf{J}(k)^T + \mathbf{K}(k)\,\mathbf{R}(k)\,\mathbf{K}(k)^T, && (2.22)
\end{aligned}$$

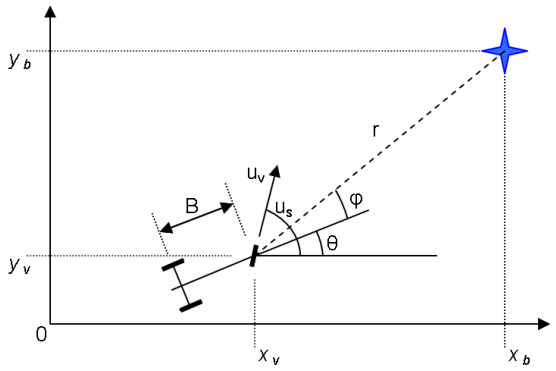

Figure 2.4: Notation used to describe the platform and beacons. The platform is modelled as a steered tricycle, and the blue star represents a point beacon being observed by the sensor mounted on the centreline of the front wheel.

where $\mathbf{K}(k)$ is the Kalman gain and $\nu(k)$ is the innovation.

2.4.1 Example Platform Model

There are a number of types of platforms on which SLAM is performed, including wearable computers with head-mounted sensors, robots and cars. For large outdoor environments, a car-like vehicle is usually used. This is more straightforward to implement compared to head-worn systems. In the 2D experiments used in this thesis, we use a fundamental steered bicycle model [58], which receives the control inputs speed $u_v(k)$ and steer angle $u_s(k)$ at each time step, with associated errors $v_v(k)$ and $v_s(k)$ as shown in Figure 2.4. These are noisily measured with zero mean and standard deviations σ_v and σ_s. The state transition function F from (2.4) is

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{v,k} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{v,k-1} + (u_v(k) + v_v(k)) \triangle t \begin{bmatrix} \cos(\varphi(k)) \\ \sin(\varphi(k)) \\ \frac{1}{B}\sin(u_s(k) + v_s(k)) \end{bmatrix}, \tag{2.23}$$

where $\triangle t$ is the time elapsed between $k-1$ and k (50 Hz for our simulations), B is the vehicle wheelbase (which we took as 2 m), and $\varphi(k) = \theta(k-1) + u_s(k) + v_s(k)$. The estimate according to (2.12) is

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{v,k|k-1} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{v,k-1|k-1} + u_v(k) \triangle t \begin{bmatrix} \cos(\hat{\varphi}(k|k-1)) \\ \sin(\hat{\varphi}(k|k-1)) \\ \frac{1}{B}\sin(u_s(k)) \end{bmatrix}, \tag{2.24}$$

where $\hat{\varphi}(k|k-1) = \theta(k-1|k-1) + u_s(k)$. The linearised Jacobian of (2.24), $\bigtriangledown\mathbf{F}(k)$ is given by

$$\bigtriangledown\mathbf{F}(k) = \begin{bmatrix} 1 & 0 & -u_s \bigtriangleup t \sin(\hat{\varphi}(k|k-1)) & 0 & \dots & 0 \\ 0 & 1 & u_s \bigtriangleup t \cos(\hat{\varphi}(k|k-1)) & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{2.25}$$

Note how this is the identity matrix with the exception of the two non-zero vehicular elements, showing that the map is assumed static.

Assuming the vehicle initially starts with an empty map, the initial covariance of the vehicle pose, $\mathbf{P}(0|0) = \mathbf{diag}(\begin{bmatrix} P_x & P_y & P_\theta \end{bmatrix})_i$ may be set to $\mathbf{0}$, in which case the map and vehicle poses will be estimated relative to this starting pose.

The process noise covariance $\mathbf{\Sigma}(k)$ in (2.13) is assumed to be

$$\mathbf{\Sigma}(k) = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_s^2 \end{bmatrix}. \tag{2.26}$$

The vehicular component of the linearised process model with respect to the control inputs, $\nabla\mathbf{G}_v(k)$ is

$$\nabla\mathbf{G}_v(k) = \triangle t \begin{bmatrix} \cos(\theta + u_s) & -u_s \sin(\theta + u_s) \\ \sin(\theta + u_s) & u_s \cos(\theta + u_s) \\ \frac{1}{B}\sin(u_s) & \frac{u_s}{B}\cos(u_s) \end{bmatrix}, \tag{2.27}$$

where θ is $\theta(k-1)$ and u_s is $u_s(k)$. The remaining elements in $\nabla\mathbf{G}(k)$ are $\mathbf{0}$ as the environment is static. It should be noted that $\nabla\mathbf{G}_v(k)$ represents the discrete-time process noise covariance, which corresponds to the integrated effect of the continuous-time process noise over $\triangle t$ [38]. We have not had to use stabilising control noise from (2.13) in our simulations, as our true system uses ideal process models and the noise characteristics are exactly known; thus $\mathbf{Q}_s = \mathbf{0}$.

If a new beacon is to be initialised into the map, provided it is observable at the time it is first observed, its estimate is given by applying the *inverse observation function*

$$\mathbf{x}_i = \hbar[\mathbf{x}_v(k), \mathbf{z}_i(k), \mathbf{w}(k)]. \tag{2.28}$$

The new mean and covariance become

$$\hat{\mathbf{x}}'(k|k) = \begin{bmatrix} \hat{\mathbf{x}}^T & \hbar[\hat{\mathbf{x}}_v, \mathbf{z}_i(k), \mathbf{0}]^T \end{bmatrix}^T_{k|k}, \tag{2.29}$$

$$\begin{aligned} \mathbf{P}_i &= \nabla\hbar_i^x \mathbf{P} \nabla^T \hbar_i^x + \nabla\hbar_i^w \mathbf{R} \nabla^T \hbar_i^w \\ \mathbf{P}'(k|k) &= \begin{bmatrix} \mathbf{P} & (\nabla\hbar_i^x \mathbf{P})^T \\ \nabla\hbar_i^x \mathbf{P} & \mathbf{P}_i \end{bmatrix}_{k|k}, \end{aligned} \tag{2.30}$$

Table 2.1: Vehicle parameters used in the simulations.

Parameter	Symbol	Value
Starting uncertainty	$\begin{bmatrix} P_x & 0 & 0 \\ 0 & P_y & 0 \\ 0 & 0 & P_\theta \end{bmatrix}_{0\|0}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} .10^{-5}\ (\mathrm{m}^2, \theta^2)$
Vehicle speed	u_v	14 m/s
Steer noise	σ_s	3°
Speed noise	σ_v	0.5 m/s
Odometry time period	$\triangle t$	0.02 s (50 Hz)
Vehicle wheelbase	B	2 m

where ∇h_i^x and ∇h_i^w are the Jacobians of $h[\cdot]$ with respect to the vehicle and observation noise respectively [61]. In some cases this can be computed using measurements collected from several time steps [126].

Certain desirable sensors in SLAM (most notably a camera) are only *partially observable*, in which case initialising a beacon into the map is more complicated. In this case $\mathbf{z}_i(k)$ consists only of a bearing observation, so the estimate of $\mathbf{x}_i$ in the range (depth) direction has effectively infinite uncertainty. In this case applying (2.29) and (2.30) will give a badly conditioned estimate which can result in filter divergence and failure. Thus beacon initialisation in bearing-only SLAM needs to be handled explicitly.

Having described EKF-SLAM in more detail, we will now demonstrate that EKF-SLAM suffers from defects that prevent it being an effective solution to the SLAM problem. We will do this through a simulation representing the scenario we are interested in this research; that of a vehicle performing SLAM in an urban environment (in this case representing a part of London), described below.

2.5 Simulation Conditions

We investigate the contribution of a prior map to SLAM using Monte Carlo simulations (consisting of 100 runs each) on a simulated planar environment with a configuration similar to that which may be encountered in a real life urban area. The map and the part of London it represents is shown in Figure 2.5. The trajectory consists of a large loop which a vehicle traverses at 14 m/s, loop closure occurring at approximately 4300 s, and is 1400 m long. Occlusions are modelled by rectangular regions. A number of 2D point beacons are scattered around the environment, however most initially lie on the building walls, with an average of 1 beacon per 5 m of wall.

The line segments represent building walls that we will use later in chapter 3 as the basis of our prior information.

The vehicle simulation parameters are given in Table 2.1, and those of the range-bearing sensor in Table 2.2. The range-bearing sensor specifications are within the capability of modern LIDAR scanners, such as the SICK LMS 291.

Figure 2.5: Left: the simulated environment, units in m. The dashed line shows the trajectory, which starts near the centre (where the vehicle can be seen making observations) and completes an anti-clockwise loop before ending at the top. There are 516 beacons that lie on the building walls (they are shown as crosses), and 164 other beacons (shown as $\times$'s). The dashed rectangles show occlusion boundaries. Right: the simulated environment is based on a part of London south-east of the UCL campus, where most of the building walls close to the vehicle trajectory are modelled. (Background image source: www.multimap.com)

Table 2.2: Range-bearing sensor parameters for the simulations.

Parameter	Symbol	RB sensor values
Maximum range	r_{max}	50 m
Maximum sweep	ϕ_{max}	$\pm 90°$
Range error	σ_r	0.01 m
Bearing error	σ_ϕ	$1.15°$
Update frequency	n/a	50 Hz

As EKF-SLAM scales cubically with the number of beacons and our Matlab implementation does not used advanced map management techniques such as submapping with the Compressed EKF [48] or Thin Junction Tree Filters [97], for computational reasons we use a sliding window, discarding all beacons not seen within 5 s (250 time steps), with the exception of the beacons in the loop closure region (seen in Figure 2.5). We manually selected beacons to stay there as we know that loop closure will occur there, however a more advanced implementation would not need to do this as all beacons would be kept.

Although we could have used such advanced map management techniques, in this thesis we restrict ourselves to considering the performance and error propagation under a single global coordinate frame. However the techniques developed in this thesis would be usable with such methods.

2.6 Example of EKF-SLAM Problems

For the simulations performed in this thesis we introduce a consistency metric, similar to that used by [14], intended to show how many of the Monte Carlo runs exhibit a given degree of inconsistency. The metric is intended to capture the probability that the pose error for a run would remain within a given bound, thus allowing the filter to successfully gate with prior map features and close loops. Traditionally the average NEES [11] is used to indicate filter consistency, however this does not account for the distribution of inconsistency over runs. For instance the average NEES of a filter that is almost consistent over 99% of runs, and very optimistic over 1% may appear similar to a filter where 99% of the runs are moderately inconsistent, whereas the former may presumably be preferable.

For a particular MC run to be considered successful with regard to a certain upper confidence bound, we require the NEES for the vehicle pose to remain within that bound for at least 95% of the run. As we increase the confidence bound, we would expect more runs to be successful; the number of runs that pass each confidence bound threshold gives an indication of the inconsistency severity over the runs. We have found that the EKF is sufficiently inconsistent that we express the confidence bound in terms of standard deviations of the Gaussian distribution, rather than percentages. For instance, 99.7% of the mass of a Gaussian distribution is within 3σ of the mean. Thus for a consistent Gaussianly-distributed estimate we would expect almost all the runs to remain within the 3σ bound for at least 95% of the run. Although the state estimates are almost certainly not Gaussianly distributed, the vehicle pose dimension is always 3 in all cases and thus we use this to compare the algorithms in relative terms.

To illustrate the consistency problem with SLAM, consider the vehicle in Section 2.4.1 performing SLAM using the EKF on the map shown in Figure 2.6. Data association is assumed known. The results are presented for the average of 100 Monte Carlo runs.

Figure 2.7 shows that there is almost a 40% chance that during a run the actual pose error would exceed the 6σ bound of its estimated error for more than 5% of the run (in total), which by our criterion indicates that the uncertainty estimate is highly optimistic compared to the actual error, and thus the filter is highly inconsistent.

Figure 2.6 shows that unsurprisingly all of the beacons lie outside of their 3σ estimate bounds; the 3σ bound is usually used as (for a consistent filter) the true state would almost always lie within it. Although this example only shows an EKF-based system, other SLAM methods also exhibit inconsistency

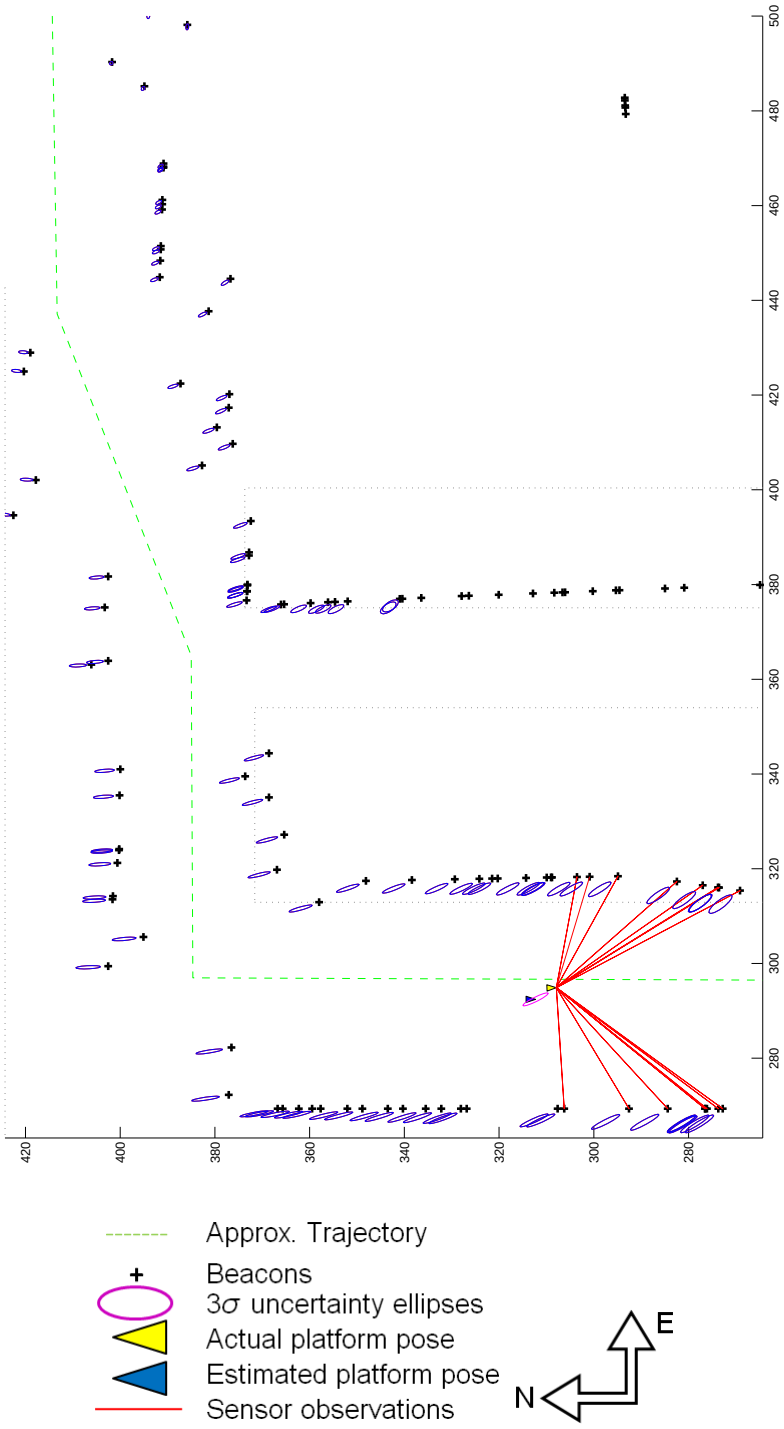

Figure 2.6: Map showing the growth of inconsistency in a run of our simulated scenario of a platform performing SLAM. The 3σ covariance ellipses represent the map estimate, and the crosses the actual positions of the features. The inconsistency is clearly illustrated by the actual feature positions lying outside of their expected uncertainty ellipses.

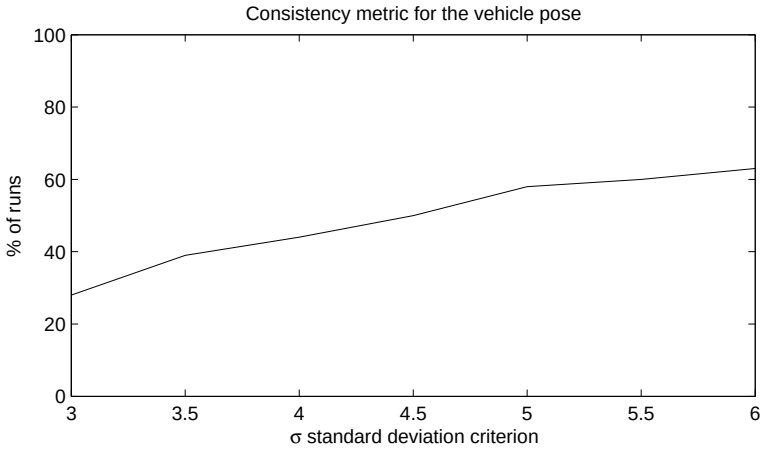

Figure 2.7: Consistency metric from our SLAM simulation, indicating for what percentage of runs the vehicle pose was within a given standard deviation criterion for at least 95% of the run. For a consistent algorithm we would expect almost all the runs to lie within the 3σ bound, however less than 30% of the runs do, showing that the filter estimate is highly inconsistent.

[121, 8].

Figure 2.8 shows the absolute errors in the vehicle pose (x, y and orientation), along with its 3σ uncertainty bound. The run appears to be consistent until 3000 time steps, when the x and orientation errors increase without a corresponding increase in uncertainty. The filter underestimates its error and is thus inconsistent.

While the above examples demonstrate the inconsistency problem with EKF-based SLAM, a number of other methods, some of which we will discuss below also suffer from inconsistency and other problems of their own. Thus although some authors consider the SLAM problem to be effectively solved [32], there are a number of major problems that suggest that this is far from the case in all but a small number of applications. The number of approaches found in the literature, together with the lack of practical demonstrations in difficult (prolonged large, outdoor) environments suggests that there is still much work to be done before SLAM can be truly considered solved. Fundamentally there are a number of challenges at the heart of the SLAM problem that make it difficult to solve.

2.7 Why is SLAM Difficult?

There are a number of aspects of the SLAM problem that make it difficult to solve.

A consistent solution to the SLAM problem requires that a complete distribution over the vehicle and all beacon estimates is maintained [61, 18, 32]; not doing so will lead to an erroneous map and inconsistent vehicle estimate [18]. The computational cost of updating the correlations between beacons quickly becomes computationally intractable for maps with a moderate to large number of beacons [85, 25, 32]. For a fairly detailed 3D map, the number of dimensions can be very large [119, 43].

Robot motion is subject to errors (for instance due to wheel slippage) which ideally would be avoided by staying in one place [117]. However very few environments can be mapped from a single

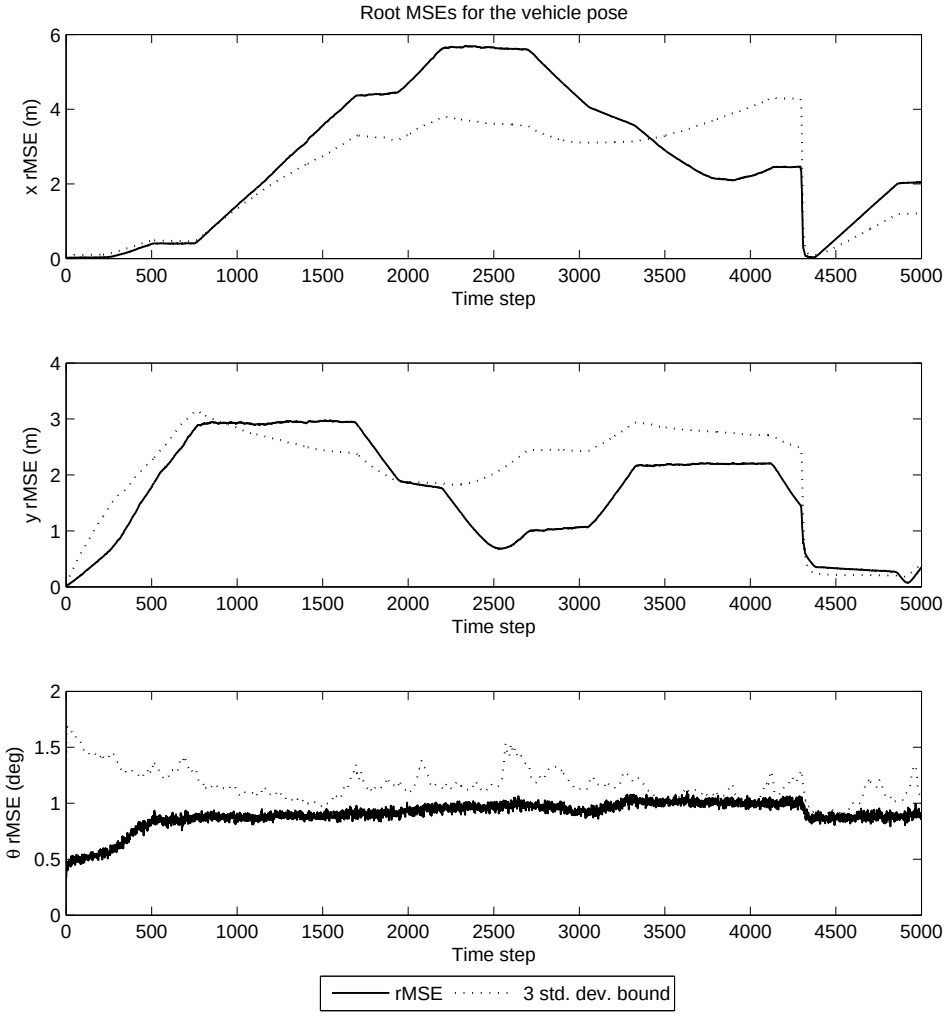

Figure 2.8: Root MSEs and 3σ standard deviation bounds for the vehicle pose from our SLAM simulation, showing the onset of inconsistency in the latter stages of the run.

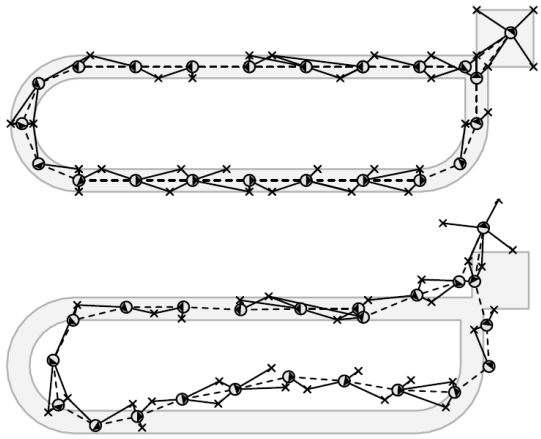

Figure 2.9: A typical SLAM run using the EKF from [41]. a) The true robot map b) The estimated map. Nonlinearity causes the map to become steadily distorted.

position; sensors have limited range (as with SICK scanners) and observability (as in the case of cameras) and obstructions block beacons [119]. SLAM can be even harder for a platform not in control of its motion, such as a wearable computer. Thus process noise from platform motion is inevitable.

Because the sensor is mounted on a platform that has a pose error, any measurements and thus the map produced will inherit this error. Furthermore, unlike the measurement noise, whose effect reduces with multiple observations, the platform noise is always present. Thus SLAM has a specific uncertainty structure, described as *certainty of relations despite uncertainty of positions [43]*. In this structure, the relations between nearby landmarks are known precisely even though the position of the landmarks is highly uncertain. These systematic and correlated map errors are a complicating factor in robotic mapping, as errors at previous time steps accumulate and cannot be removed [119, 117].

The nonlinearity of the platform and motion models further complicate SLAM, the main source of issues stemming from the robot orientation uncertainty [43]. EKF-SLAM, which linearises the nonlinear models, is particular sensitive to nonlinearities, which cause a build-up in inconsistency and map distortion as the run progresses. An example of this from [41] is shown in Figure 2.9. There have been attempts to maintain consistency by inflating the covariance after each update, however it is difficult to determine the degree of inflation required and convergence may not occur [102].

Some consider the data association problem (also known as the correspondence problem) to be the most difficult problem in SLAM [15, 119]. This is the problem of determining whether an observable beacon is new or has been seen before (and if so, which beacon it is). It also encompasses the problem of distinguishing between beacons which look alike [114]. In the latter case, the algorithm must distinguish which beacon it is seeing. Whilst initialising a previously seen beacon as a new one is undesirable but relatively benign, incorrectly identifying a beacon as another can result in an inconsistent estimate or even catastrophic failure of the algorithm [34, 43, 117].

Given these challenging aspects of the SLAM problem, a number of approaches have been proposed, including extensions to the EKF and entirely different approaches. Let us start by considering an

extension to the EKF that seeks to reduce the errors in EKF-SLAM demonstrated above by considering higher order terms in the linearisation of the observation models.

2.8 Higher Order Filters

Because the EKF performs a first-order linearisation of the nonlinear system, the output distribution is often not adequately described by the mean and covariance alone. This can introduce large errors in the true posterior mean and covariance, degrading the quality of the filter estimates. A strategy to reduce this is to capture the effect of the higher order terms of the nonlinear system on the posterior mean and covariance.

The Unscented Kalman Filter (UKF) [59] uses the unscented transform to obtain a posterior estimate that accounts for higher order terms. It does not require Jacobians to be computed, and thus is particularly suited to complex models where Jacobians are difficult to compute. However, the UKF is computationally expensive compared to the EKF, thus an analytical method is preferable where the nonlinear models are relatively straightforward, as is the case with the models in this thesis.

We use the alternative approach of Truncated Second Order filters [56], which capture the second-order statistics. Although these require the Hessian to be computed, the sparseness of the observation Jacobian $\nabla\mathbf{H}$ means that the Hessian tensor $\mathbf{M}$ is also sparse. The computational form of the Second Order filter update is

$$\begin{aligned}
\nu(k) &= \mathbf{z}(k) - \mathrm{h}(\hat{\mathbf{x}}_p, \mathbf{0})^T, && (2.31)\\
\mathbf{S}(k) &= \left[\nabla\mathbf{H}(k)\,\mathbf{P}(k|k-1)\,\nabla\mathbf{H}^T(k) + \mathbf{R} + \frac{1}{2}(\mathbf{M}(k)\,\mathbf{P}^2\,\mathbf{M}(k))\right]_k, && (2.32)\\
\mathbf{K}(k) &= \mathbf{P}(k|k-1)\nabla\mathbf{H}^T(k)\,\mathbf{S}(k)^{-1}, && (2.33)\\
\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k) \times [\nu(k) - \frac{1}{2}(\mathbf{P}(k|k-1)\,\mathbf{M}(k))], && (2.34)\\
\mathbf{J}(k) &= \mathbf{I} - \mathbf{K}(k)\,\nabla\mathbf{H}(k), && (2.35)\\
\mathbf{P}(k|k) &= \mathbf{J}(k)\,\mathbf{P}(k|k-1)\,\mathbf{J}(k)^T + \mathbf{K}(k)\,\mathbf{R}(k)\,\mathbf{K}(k)^T. && (2.36)
\end{aligned}$$

where $\mathbf{S}$ is the innovation covariance and ν is the innovation. This update is very similar to that of the EKF, with the addition of a correction to ν and $\mathbf{S}$.

The structure of the Hessian, showing its sparsity, for a beacon b is

$$\mathbf{M}^b(k) = \begin{bmatrix}
\mathbf{M}_{vv} & \mathbf{0} & \dots & \mathbf{M}_{vb}^T & \dots & \mathbf{0} \\
\mathbf{0} & 0 & \dots & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\mathbf{M}_{vb} & 0 & \dots & \mathbf{M}_{bb} & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\mathbf{0} & 0 & \dots & 0 & \dots & 0
\end{bmatrix}_k^b, \qquad (2.37)$$

where $\mathbf{M}_{vv}$ and $\mathbf{M}_{bb}$ are the vehicle and beacon derivatives respectively, and $\mathbf{M}_{vb}$ are the mixed derivatives. The equations are relatively straightforward to implement using numerical or symbolic differenti-

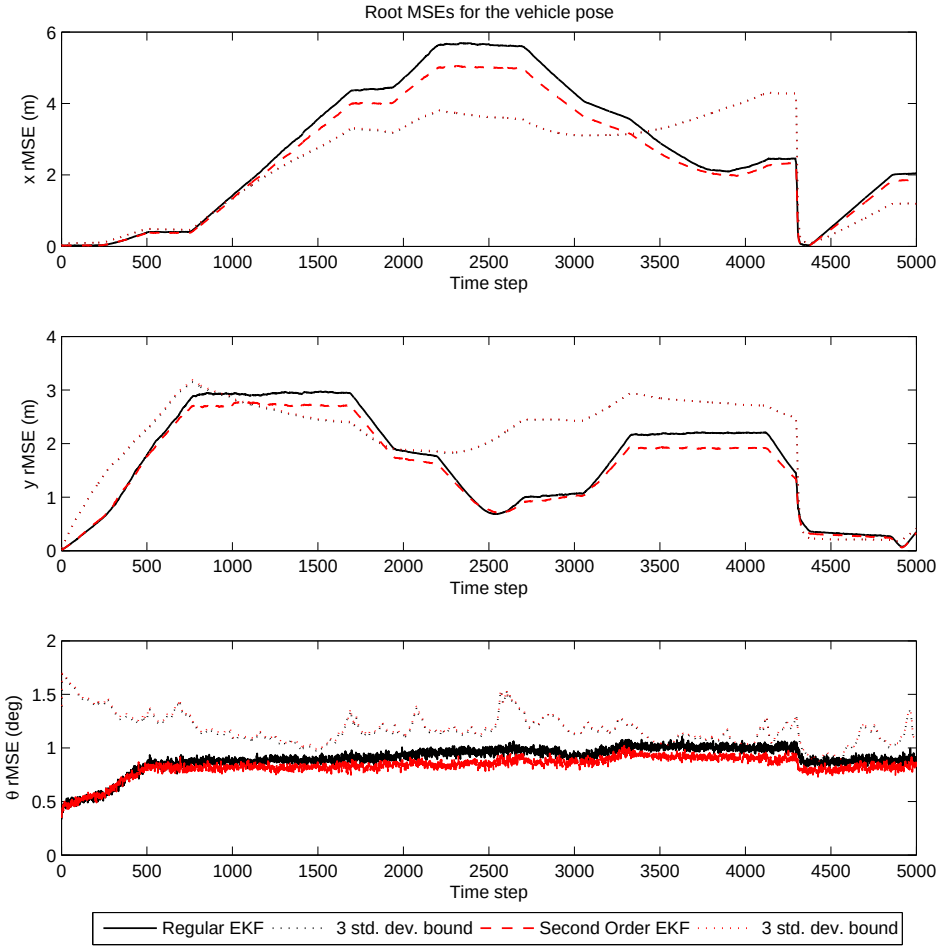

Figure 2.10: Comparison of the vehicle pose root MSEs and 3σ variances for the EKF and second order filter. The Second Order filter gives a small accuracy increase in this case, but does not qualitatively improve performance.

ation, and the overall computational costs remain $O(n^3)$.

Figure 2.10 shows that while there is a slight improvement, the pose accuracy for the Second Order filter is very similar to that of the EKF, showing that the using higher order filters does not solve the consistency problems of SLAM. This suggests that in many cases the greatest cause of inconsistency in EKF-based SLAM is the inability of the EKF to propagate angular errors correctly [61], rather than linearisation errors. However we do use the Second Order filter later in this thesis to reduce errors stemming from nonlinearities.

EKF-based approaches are unable to handle nonlinear process models directly, thus alternative methods that can inherently the nonlinearity have been used in SLAM with some success. The most popular of these is FastSLAM, which will be discussed in the next Section. However, as will be mentioned FastSLAM is not a consistent SLAM solution. The lack of a SLAM solution that can handle nonlinearity whilst meeting the requirements for a SLAM algorithm discussed in Section 2.2 is a moti-

vation for our approach of leveraging prior information to address these issues.

2.9 Other Incremental Methods

There are a number of other methods used for performing SLAM, the most popular being FastSLAM and sparse extended information filters (SEIFs). FastSLAM is more efficient than the EKF and is able to handle nonlinear process models. SEIFs are similar to the EKF, and express the correlations between beacons in a form that allows for a significant reduction in the computational and storage costs compared to the EKF.

2.9.1 FastSLAM

FastSLAM is a particle filter based approach to SLAM. Particle filters perform sequential Monte Carlo estimation based on a particle representation of probability densities [101]. FastSLAM exploits the above-mentioned structure of SLAM, where correlations in the uncertainty among different map features arise only through robot pose uncertainty [117]. There are two versions of FastSLAM, 1.0 and 2.0, of which FastSLAM 2.0 is the more efficient [6, 117].

FastSLAM is a trajectory-oriented, rather than pose-oriented method [7], estimating a posterior over robot paths rather than just momentary poses [117]. Using Rao-Blackwellisation, the joint state is represented in a factored form [6] such that the posterior over paths and maps is decomposed into a number of recursive estimators; one for the vehicle and one for each beacon [117]. The form of (2.7) is

$$p(\mathbf{x}_v(K), \mathbf{x}_b|\mathbf{z}(K), \mathbf{u}(K)) = p(\mathbf{x}_b|\mathbf{x}_v(K), \mathbf{z}(K))\, p(\mathbf{x}_v(K)|\mathbf{z}(K), \mathbf{u}(K)), \tag{2.38}$$

where $K = 1 : k$, $\mathbf{x}_v(k)$ is the vehicle pose at time k ($\mathbf{x}_v(1 : k)$ being the trajectory), $\mathbf{x}_b$ is the map of beacons, $\mathbf{z}(1 : k)$ is the sequence of observations and $\mathbf{u}(1 : k)$ is the sequence of control inputs. The trajectory posterior estimate $p(\mathbf{x}_v(K)|\mathbf{z}(K), \mathbf{u}(K))$ is modelled by a set of particles, while the beacon estimate $p(\mathbf{x}_b|\mathbf{x}_v(K), \mathbf{z}(K))$ consists of a set of independent distributions, one for each particle. The trajectory estimate is calculated using a particle filter, while the beacons are estimated analytically by low-dimensional EKFs (each EKF estimates a single beacon) [117].

FastSLAM has a number of advantages over the EKF formulation. Because the map is represented as a set of independent Gaussians, the update complexity is linear in the number of particles [6], and it inherently supports nonlinear motion models. FastSLAM also allows for multiple hypotheses, maintaining posteriors over multiple data associations rather than just the most likely one [117].

However FastSLAM has been shown to be inconsistent, degenerating over time, regardless of the density of beacons in the environment and the number of particles used [8]. This is due to the nature of particle filters, where the approximation error accumulates over time, requiring a corresponding increase in the number of particles. With a finite number of particles, the variance of sample weights increases over time, resulting in all samples but one possessing negligible weight. Resampling is performed to offset this, however this will only allow for consistent estimation provided that any error is forgotten exponentially over time (which does not occur in SLAM). Consequently, whenever a particle is removed

an entire pose history and map hypothesis is permanently lost, and the remaining particles lose track independence and landmark estimate diversity [8].

Because particle filters do not modify the robot trajectory represented by the particles chosen, nor back-propagate the error along the loop, FastSLAM generally requires many particles to be able to perform loop closing [43].

An alternative to FastSLAM that is very similar to the EKF, but reduces its computational cost for updates is the Sparse Extended Information Filter (SEIF).

2.9.2 Sparse Extended Information Filters (SEIFs)

SEIFs are based on the Extended Information Filter (EIF). The EIF is the dual of the EKF, being algebraically identical and therefore subject to the same assumptions as those underlying the Kalman filter [114, 77]. The main difference between the two is that whereas the EKF represents the map and vehicle by their mean and covariances, the EIF represents them by an information matrix and information vector (called the *canonical parameterisation*) [114]. The information matrix is the inverse of the covariance matrix. While the EIF and EKF are computationally equivalent [43, 114, 121] and both incremental, they differ in the complexity of their process and measurement updates. The process update for the EKF is relatively simple; the measurement update complex. The situation is reversed for the EIF [114].

The EIF has a number of advantages over the EKF. The EIF also tends to be more numerically stable than the EKF, though Anderson and Moore [4] note that under certain conditions there could be numerical difficulties when using the information filter equations, for instance when a feature is known with high accuracy, in which case the information form tends towards infinity. The EIF is particularly well suited to SLAM using multiple robots, as information can be integrated in an arbitrary order with arbitrary delays, and in a decentralised manner [114].

The EKF remains vastly more popular due to a number of limitations of the EIF [114, 121]. The EIF needs to recover a state estimate in the update step, which requires a matrix inversion. In general, the complexity of the EIF is cubic in the size of the state space [121]. For high dimensional state spaces, the EIF is thus considered computationally inferior to the EKF [114].

For these reasons, the standard implementation of the EIF is not used. Instead, sparsity in the structure of the information form is exploited in an algorithm called the Sparse EIF (SEIF). Thrun [121] defines an information matrix as being sparse if the number of links to the robot and to each feature of the map is bounded by a constant independent of the number of beacons. The elements of the information matrix can be considered as links between the locations of different beacons [77]. The stronger the link, the larger the element in the information matrix. Robot motion has the effect of linking features that were previously linked only through the robot. When normalised, the information matrix tends to be naturally sparse, consisting of a small number of strong links between nearby beacons, and a large number of weak (near-zero) links between the rest. Frese offers a proof for the sparsity of the information matrix in [40]. The strength of the links is related to the distances between the beacons [77]. The sparsity of SLAM information matrices is shown in [43]. The SEIF therefore maintains the strong links, and sets the weak (near-zero) links to zero. The benefit of this form is that storing a sparse information

matrix requires linear space, and both the process and observation updates can be performed in constant time regardless of the number of beacons (provided the mean is available) [77, 121]. Thrun [121] has performed experimental runs with the SEIF, and has found that compared to the EKF, the SEIF runs twice as fast and consumes less than a quarter of the memory, at a penalty of being slightly less accurate.

In enforcing sparsity, the resulting information matrix is only an approximation of the original, whose quality depends on the strength of the removed link [77]. Some methods, such as [121] sacrifice consistency in enforcing sparsity, however as mentioned this is undesirable. Recently, exactly sparse EIFs (ESEIFs) have been developed which do not suffer from this, however the estimate is conservative [123].

Unfortunately the similarity of the EIF to the EKF means it suffers from many of the same problems, including linearisation and angular errors and the inability to maintain multiple data association hypotheses.

If the requirement for incremental (online) operation is relaxed, a number of methods which aim to find the maximum likelihood (ML) solution can be used [43]. The maximum likelihood estimate is the joint platform/map which has the largest probability of causing the control and sensor observations made.

2.10 Maximum Likelihood

Unfortunately finding a maximum likelihood map is computationally challenging because it involves searching in the space of all maps, whose size is generally very large, and thus is not practical for incremental SLAM [41, 120]. The global update complexity is $O((n+p)^3)$, where n is the number of beacons and p the number of platform poses [41, 43], compared to $O(n^3)$ for the EKF (which can be considered to be equivalent to maximum likelihood estimation over a single time step, where the previous time steps have been marginalised out [105]).

The maximum likelihood estimate can be considered in the context of a graph-based formulation of SLAM, termed GraphSLAM [122]. GraphSLAM is based on the intuition that the posterior of full SLAM forms a sparse graph. In this graph each node represents a platform pose, and edges by the associations between the observations (odometry and sensor) made at these poses. GraphSLAM is based on the work of Lu and Milios [78], which estimates the platform poses at which range scans (such as LIDAR point clouds) were made, using the pose relations as constraints.

If the observation errors between poses are independently Gaussian with known covariance, the maximum likelihood estimate corresponds to the minimum of the sum of nonlinear quadratic constraints corresponding to the sensor observations (given the data association parameter $\mathbf{c}$) L, and odometry observations G,

$$L(k) = (\mathbf{z}(k) - \mathrm{h}(\mathbf{x}, \mathbf{c}))^T \ \mathbf{R}(k)^{-1} \ (\mathbf{z}(k) - \mathrm{h}(\mathbf{x}, \mathbf{c})) \,, \tag{2.39}$$

$$G(k) = (\mathbf{x}_v(k) - \mathrm{f}(\mathbf{x}_v(k-1)))^T \ \mathbf{Q}(k)^{-1} \ (\mathbf{x}_v(k) - \mathrm{f}(\mathbf{x}_v(k-1))) \,, \tag{2.40}$$

$$\hat{\mathbf{x}}_v(1:k), \hat{\mathbf{x}}_b = \underset{\mathbf{x}_v(1:k), \mathbf{x}_b}{\arg\min} \sum_{t=1}^{k} L(t) + \sum_{t=1}^{k} G(t), \tag{2.41}$$

where $\mathbf{x}_v(1:k)$ is the set of vehicle poses and $\mathbf{x}_b$ is the map (which is assumed static). This system can be solved by techniques such as the conjugate gradient method [122, 69].

Expectation Maximisation (EM) [31] can also be used to compute the maximum likelihood map, along with the data associations. This uses hill climbing in likelihood space to maximise the expectation over the joint log likelihood of the observations and the platform's path [119]. It consists of two alternating steps; an expectation and a maximisation step. In the expectation step probabilistic estimates of the robot's locations at various points in time are estimated based on the current best available map. In the maximisation step, a maximum likelihood map is estimated (along with the data association) based on the locations computed in the expectation step [120]. Iterating both leads to a refinement of both the platform poses and the map.

EM has the advantage of being one of the best solutions to the data association problem [119, 120], as the unknown associations are estimated along with the map in the maximisation step.

However EM has the disadvantages of being a numerical approach, so it does not maintain a full notion of uncertainty [119], and tends to converge to a local rather than global maximum.

For our purposes we will not be considering smoothing methods as they are more complex than filtering [4] and because we require a method that will run incrementally. While batch processing provides the most accurate and robust solution to any estimation problem where off-line processing is satisfactory, incremental SLAM operation requires estimation that can run in constant time [25]. While there has recently been an increasing body of promising research in applying smoothing methods in real-time [62, 104, 64], we consider smoothing methods to be too immature for timely implementation of the probabilisitic frameworks in this thesis, and will thus use filtering, with the intention of possibly applying our methods to smoothing approaches in the future.

2.11 Main SLAM Research Centres

There are several major research centres for SLAM, focusing on varying SLAM applications and sensor types:

- The ARC Centre of Excellence for Autonomous Systems, which includes groups at the ACFR (Australian Centre for Field Robotics), University of Technology Sydney (UTS) and University of New South Wales (UNSW) is a major hub for SLAM research and experimental application, hosting a number of leading researchers including Hugh Durrant Whyte, Tim Bailey (who has analysed the consistency of FastSLAM and the EKF), Eduardo Nebot and Gamini Dissanayake. The popular Victoria Park data set [47] was collected at the UTS.

- The University of Freiburg has a strong robotics group performing 3D SLAM using small mobile robots equipped with 3D laser scanners. The main technique used is scan pose alignment, the direct alignment of dense point clouds produced by 3D laser scanners [91, 92]. Recently the group has started to look at integrating prior information into GraphSLAM [69]. Prominent researchers include Wolfram Burgard and Andreas Nüchter.

- The Robot Vision Research Group at Imperial College, led by Andrew Davison has a strong emphasis on monoSLAM (single camera SLAM), generally for hand-worn devices [87, 25, 27, 29]. They have traditionally used EKF-SLAM with a sparse set of features for this purpose, and developed the Inverse Depth parameterisation for representing large depth uncertainty using a single EKF state [87]. Recently the group has started to focus on a smoothing rather than filtering approach to monoSLAM [110].

- The Robotics, Perception and Real Time group at the University of Zaragoza perform research in several SLAM areas, including monoSLAM. In particular, Javier Civera and José Montiel have collaborated with the Robot Vision Research Group at Imperial College on several key monoSLAM publications [87, 110]. Other key researchers include José Neira, Juan Tardós and José Castellanos, and among them have introduced the popular Joint Compatibility Branch and Bound (JCBB) data association method [88], and the SPmap representation for uncertain geometry in maps [19].

- The Oxford Mobile Robotics group, led by Paul Newman, have a number of small robots with which they perform SLAM in large outdoor environments, mainly using stereo vision sensors. Our experiment in chapter 6 uses data from one of the robot experiments performed by this group. The group emphasises robot perception and understanding for robots operating autonomously for long periods of time, and to this end has developed several vision based algorithms for semantic labelling [98] and loop closing [24], and techniques for accurate *local* (rather than global) localisation and mapping, namely relative bundle adjustment [104].

- Stanford University's Artificial Intelligence Lab (SAIL), whose Robotics and Machine Learning group is led by Sebastian Thrun, is active in probabilistic robotics, the application of probabilistic techniques to robotic problems (including SLAM). Thrun has overseen the development of two of the SLAM algorithms discussed above; FastSLAM [85, 84] and SEIFs [121, 77], and has also demonstrated a number of robotic systems including robotic tour guides [113] and a robot for mine mapping [115]. The group also won the DARPA Grand Challenge with the "STANLEY" autonomous car [116], and came second in the DARPA Urban Challenge with their car "Junior".

2.12 Conclusion

While the methods above are all designed to improve certain aspects of the SLAM problem, such as computational cost, they are generally used under the traditional assumption of there being no prior information about the environment, and are thus limited to the platform observations. This means that

regardless of how good the algorithm is, it cannot mitigate errors in the motion and observation models, and could still be subject to significant drift with unbounded error, especially on long trajectories with no loops. In addition, each of the aforementioned SLAM methods only adequately addresses some of the SLAM problems, often without addressing or seeing reduced performance in the others.

Thus in the next chapter we propose an alternative approach to improving SLAM, where rather than trying to improve the performance of the algorithms on the standard SLAM problem, we seek to improve SLAM by leveraging any prior information known about the environment.

We have chosen to use the EKF as the SLAM algorithm on which we will implement our techniques for several reasons.

Firstly, the EKF has mathematically rigorous roots. The EKF is based on the Kalman Filter, which under known conditions (linear systems and representative uncertainties) is known to be the optimal MMSE (minimum mean square error) estimator. Although these results do not necessarily hold for the EKF because of the use of nonlinear motion models, it is an open problem of how to extend the Kalman Filter to such models in a way that guarantees optimality, and progress in this area would be directly applicable to EKF-SLAM and the techniques in this thesis. The extensive use of the EKF in the literature suggests that heuristically, the EKF is still effective even though the optimality results do not apply when nonlinear motion models are used.

Secondly, the EKF explicitly encodes a probabilistic representation (the mean and covariance) of the state, maintaining the cross terms (covariances) of all the variables in the state. Thus it is relatively straightforward to extract the mean and covariances of the marginal distributions of the platform pose and map, required for visualisation, performance analysis and data association. This is in contrast to the main alternatives: FastSLAM, which uses a particle filter representation whose performance can depend on the quality of heuristic resampling strategies, and SEIFs, where the marginal distributions can be computationally expensive to recover.

Thirdly, the EKF is a filtering solution (and is in fact equivalent to smoothing over a single time step [105]), which we have chosen instead of smoothing for the work in this thesis as we desire a base SLAM technique that has been widely implemented, does not have the complexity of smoothing over multiple time steps, and which can conceptually work in real time. Smoothing approaches to SLAM are relatively immature compared to EKF-SLAM, thus it is unclear how easy it would be to implement such a technique for the work in this thesis.

In the following chapter, we give an overview of the state of the art in the use of prior information in SLAM, then formulate an interpretation of maps (of which the SLAM state is an example) and the environment which facilitates the development of a probabilistic model for exploiting information between maps, which we then implement using EKF-SLAM with prior information consisting of a prior map.

Chapter 3

Using Prior Information in SLAM

As discussed in the previous chapter, despite its conceptual simplicity, SLAM presents many theoretical and practical problems, and of the many solutions proposed none meet all the nominal optimality, consistency and performance requirements of a functional SLAM solution.

However, almost all SLAM research makes the assumption that there is no prior information about the environment, and that when there is such information it uses the same representation as the SLAM state. These assumptions are unlikely to be true in practice. Even in civilian applications, high-quality prior information about urban environments can be readily obtained from many sources. These include cartographic data (historical maps, street maps, GIS), human intelligence (sketch maps or reports), and data collected from other platforms (such as other UAVs, satellites, or even unattended ground sensors). For example, the Ordnance Survey MasterMap aims to map all building structures with an accuracy of 1m. Even the surface of Mars has been mapped to an accuracy of 1.5 m per pixel in some areas [81].

Given that the prior information is collected by a disparate set of agents, each with their own sensing and signal processing capabilities, each might not represent the same environment in the same manner. For example, consider the situation shown in Figure 3.1: the platform is equipped with a sensor such as a laser range finder or a camera, and picks up point features on building walls from its vantage point on the ground. The camera on the satellite sees the same environment from above, from which the edges of building walls are detected. Thus the platform (satellite vs. vehicle), sensors used (laser scanner vs. camera), vantage point (ground-level vs orbit) and feature types (points vs. line segments) are different, with different error characteristics. However although this is the case, it still makes a great deal of sense to explore whether this information can be used to improve the metric accuracy of SLAM, provided we explicitly account for this heterogeneity.

In this chapter we present a novel SLAM-oriented probabilistic framework for exploiting heterogeneous prior information through the relations and structure that humans semantically know to be present in the environment. Using our scenario of a vehicle driving around an urban environment where a geometric prior map exists, we show how the task of using the prior map to inform the SLAM system is non-trivial in an EKF-SLAM framework, even with known correspondences between the prior map and SLAM map, because of errors caused by the inability of the EKF to correctly handle nonlinear models as discussed in the previous chapter. To address this, we formulate a novel parameterisation for mitigating

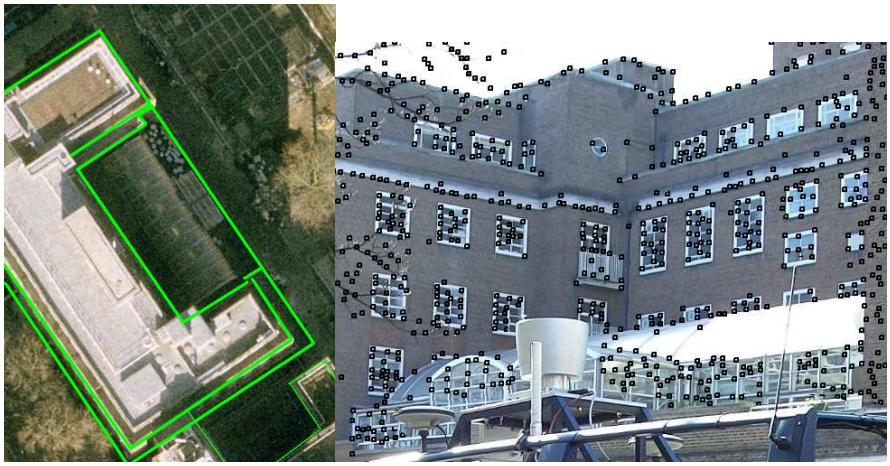

Figure 3.1: Left: satellite image with line segment features extracted by hand. Right: ground camera image with extracted Harris corner features. The images represent heterogeneous features obtained from two different views of the same building.

Table 3.1: Types of prior information used in SLAM with specific examples.

Absolute	Relational	Topological	Semantic
Beacon positions	Orthogonality [12, 102, 108]	Repetitive features [20]	Place recognition [24]
Platform pose [69]	Parallelism [1, 45, 108]	Easily partitioned [42]	
	Planar ground [47]		

these errors using information from the prior map.

Having determined how to robustly integrate a prior map in EKF-SLAM, we then show the theoretical benefits this gives on the vehicle localisation performance.

Let us begin by considering the main types of prior information that are used in SLAM.

3.1 Information Types

We have identified four main types of prior information; absolute, relational, topological and semantic, of which the latter three have been covered to some degree in the literature [102, 36, 24]. These are presented with examples in Table 3.1, and discussed in more detail below. *Absolute* (also called dimensional) information introduces external data to constrain the position of beacons in space or in relation to each other. *Relative* constraints relate to how beacons lie in relation to one another, without absolute knowledge [83]. *Topological* constraints relate to assumptions about the connectivity between places, and tend to be more subtle than absolute and relational constraints, in that they tend not apply to particular beacons but to the map as a whole. *Semantic* information exploits non-geometric features of the environment to aid in SLAM. The most obvious application is visual processing of camera data.

3.1.1 Absolute Information

Most SLAM research considers environments as being either completely devoid of any prior knowledge [119], or sufficiently well mapped to present a localisation rather than a SLAM problem [128, 45, 30]. However, most problems for which we seek to apply SLAM will lie somewhere in between these two extremes. When considering prior information in this context, there are a number of factors that need to be addressed. The amount of prior data available and its accuracy needs to be considered. In particular, how poor can the data be whilst still being useful? At what point is the data sufficiently accurate so as to fundamentally simplify the SLAM problem? The effect of using incorrect or inaccurate data (for instance obsolete maps) is also an issue. There are at least two examples in the literature where standard SLAM runs have produced maps which highlight deviations of actual buildings from their designs (the building plans were not used in the SLAM algorithm itself) [37, 119]. To date these factors have not been adequately addressed in the literature.

Such absolute prior information, when used during the SLAM process can potentially prevent the SLAM map from becoming deformed to the extent that loop closure can no longer be reliably performed, and can help to improve data association, for instance by indicating areas that are susceptible to producing unstable features; for instance features that lie on roads may correspond to moving cars, so may be avoided.

There are a number of issues with using prior information in this way, and unknowns that need to be accounted for. For instance, data can become obsolete, temporary changes can occur to a building appearance (e.g. due to scaffolding), or the data may have unmodelled errors. An example of such error is distortion caused by heat haze in maps derived from aerial imagery [33]. There is also the need for the data to be compiled and updated, bringing increased complexity to the operation of a practical SLAM system. Depending on the nature and resolution of the data, storage and data protection requirements will also have to be considered.

Guivant et al. [50] note that absolute information from sporadic GPS observations and beacons with prior position estimates can be integrated into EKF SLAM, and consider the numerical and linearisation effects associated with incorporating such observations. However they do not consider prior information in further detail, such as its origin (other than GPS), error structure and the problem of associating prior map features with the SLAM state. In addition we also consider *heterogeneous* metric maps, where the prior map may have a *different* parameterisation to those of features in the SLAM map, as this is a more general and widely applicable case.

Yun and Miura [128] use a rough map consisting of planar walls modelling buildings to perform EKF-based localisation in outdoor environments. However pure localisation does not specifically account for correlated errors in the prior map, and requires the prior map to span the entire area of operation. By integrating the prior map into SLAM, we overcome these problems.

Kümmerle at al. [69] use a prior map in the form of edges extracted from an aerial image to augment a Graph-based ([122]) formulation of SLAM, in a similar manner to Monte-Carlo Localisation ([30]). The effect of using the prior map is an increase in the accuracy of the trajectory and map produced by the

SLAM run. However errors in the prior map are treated as observation errors, and only contribute to a platform pose estimate. The method appears to be oriented towards high quality aerial imagery-derived prior maps (as opposed to other agents performing SLAM for instance), so SLAM with sparse landmarks are likely to be challenging. In addition, because the prior map errors are not estimated at the feature level, prior maps with correlated errors and effects such as missing, corrupted or spurious features are not explicitly accounted for.

Lee et al. [72] integrate a prior map consisting of road segments into a FastSLAM [85] formulation to constrain the location of a vehicle performing SLAM, significantly improving its pose estimate. We aim to be able to use prior information pertaining to the SLAM features in addition to the vehicle.

3.1.2 Relational Information

Relative constraint methods attempt to exploit correlations between landmarks due to structure in the environment. They make an assumption as to the true position of certain beacons in relation to each other based on how close the estimated beacon positions approximate a given criterion, usually co-linearity (parallelism) or orthogonality. Beevers and Huang [12] have evaluated this method using a particle filter approach, and have found that the method can yield a significant improvement in the resulting maps and a reduction in estimation error (however, consistency is not guaranteed). In addition, using a particle filter incorporating constraints allows one to use fewer particles, giving improved computational performance.

Matia et al. [102] look for orthogonality or parallel relationships between line segments extracted from laser data in indoor environments, under the assumption that such environments are mainly composed of straight walls. They obtain good results in a large (approximately orthogonal) indoor map (~650 m), and have applied the system in the *Urbano* robot tour guide and the *Guido* robot for the elderly. However, their method is tailored for use in indoor environments consisting of straight, mainly orthogonal and colinear walls, and the relationships are not inferred probabilistically but rather using approximations based on Mahalanobis distance. Thus the method is unsuitable for outdoor environments that feature randomly oriented walls and curves.

Gee et al. [44] detect planes in the environment using a 6D handheld camera, and use them to reduce the size of the state space, thus reducing storage requirements. Their method is also used indoors, however has difficulty detecting planes once the uncertainty in the beacons becomes sufficiently high.

The difficulty in applying relational constraints to beacon locations lies in the problem of deducing what kind of relationships apply between landmarks, and whether they actually apply. Existing algorithms implicitly assume a particular kind of environment, for instance an indoor rectilinear environment, however this lacks generality, and suggests that a more principled approach would have to explicitly probabilistically reason about which relative constraints apply, using probabilistic models and informed priors. Fortunately, most rectilinear environments (i.e. buildings) have been built from plans, which if available can be used to apply absolute constraints, or determine whether certain relational constraints hold (for instance by identifying the positions of flat walls for colinearity constraints).

There are other ways in which relational prior information may be leveraged, for instance in data association and vehicle pose estimation. Urban environments in particular retain some predictable fea-

tures, such as an abundance of straight lines, from which Georgiev [45] compiles a database of building facade models to perform consistent pose estimation (CPE) as part of a SLAM solution.

3.1.3 Topological Information

Topological information represents an environment as a collection of places connected together, rather than their geometric location in space. Thus a robot is able to move between places by following the connections. This is similar to the way people navigate through environments. Choset [22] describes a method for performing topological SLAM (T-SLAM) that does not perform explicit localisation, that is it does not keep track of the robot's pose. The method uses a Generalised Voronoi Graph (GVG) to capture the salient topology of the environment. The robot is thus able to plan a path from a start to end point by moving along the GVG, even if it does not explicitly know where it is in Euclidean space. By itself, topological SLAM cannot provide the geometric data that an agent may require, and only works when the environment is topologically rich, for instance in indoor environments.

Treemap is a topological SLAM method that is designed for environments with a topology that lends itself well to hierarchical partitioning. Indoor environments naturally exhibit this kind of topology, being split into areas such as rooms and corridors [42]. Prediction-based SLAM (P-SLAM) [20] attempts to infer the structure of areas it has not explored yet from areas it has seen. It exploits the repetition seen in many buildings, for instance between different floors and wings. However, these methods generally do not work in outdoor environments as they show far less predictability [45, 28, 42].

A non-geometric method for detecting loop closure is FAB-MAP [24], which applies vision techniques to camera frames independently of the map estimate to recognise whether a particular place has been visited previously. The use of a non-geometric method facilitates loop closure when the map estimate is erroneous or highly uncertain. However it is unclear how robust the method is to large variations in the pose at which two images representing the same location were taken (for instance when comparing an oblique aerial image with a ground camera image), and whether different sensing modalities could be used.

There are other methods by which topological constraints from prior information could help outdoors. For instance, using a rough street atlas a robot could count the number of roads entering a junction to determine whether it has become lost. Another method less likely to be useful in practice is to use information about whether a beacon is visible to make inferences about position.

3.1.4 Semantic Information

Until recently, semantic data from camera sensors has not been used in SLAM; rather the positions of salient points have been extracted as geometric observations and the rest of the image has been discarded. However, with the increased use of cameras as sensors for SLAM, vision techniques are now being recognised as having potential for augmenting metric SLAM. The FAB-MAP loop closure method discussed above is an example of a technique that extracts features to use in a non-geometric way, using them independently of the geometric map estimate.

There are many other ways in which vision data can help in SLAM, most notably in the data association problem. By recognising appearance signatures, which encode various aspects of the features the

platform is observing such as their texture or object classification, the system can reduce the uncertainty in associating observations with features in the state [7]. In addition, semantic information allows the platform to infer things about the environment based on where the features lie. For instance, vehicles tend to reside on roads, and bricks on buildings; vehicles are rarely found on buildings.

We would like our framework for exploiting prior information to be general enough that this kind of semantic information could be used.

3.2 Benefits of Using Prior Information

Given a geometric prior map of an environment, if the map can be used to inform the position of some beacons in a SLAM map, their error will be bounded by the error in the prior map, regardless of the platform pose error [41]. This potentially allows SLAM to be applied to very large environments, where the platform pose uncertainty might otherwise grow to unmanageable levels.

Prior information could help to manage computational complexity by allowing a means by which the utility of maintaining beacons in the state for loop closure could be evaluated. For instance, in a road network environment beacons at junctions are likely to have more value for loop closure than those on single roads, so should be checked for loop closure first, and given higher priority when deciding which beacons to remove from the state. For instance in the simulation experiments in chapter 2, we determined which beacons to maintain in the state for loop closure manually, however a prior map could inform such a process instead.

Using prior information can also make a SLAM solution more robust. If the SLAM process experiences a catastrophic fault that would otherwise go undetected (for instance the map slips by a large amount), for instance due to a data association failure or strong nonlinearities, comparison with prior information may allow the failure to be detected. For instance if a geometric prior map of building locations indicates that the platform is going through buildings, but the camera indicates it is outside, the possibility that the map has slipped or the robot has been kidnapped can be considered.

With constrained beacon locations, it is possible that catastrophic failure may be avoided or reduced. Prior constraints may provide a basis by which beacons which look alike can be distinguished, and incorrect data associations can be identified and mitigated. If failure does occur however, a degree of recovery may be possible due to the bounded position error. For instance a military UAV performing SLAM may be able to recover from errors made due to chaff or flare deployments by hostile forces.

Prior information also raises the possibility of applying negative information; information concerning where a beacon cannot lie. This would be of particular benefit when performing bearing-only SLAM [25], to reduce the large initial beacon depth uncertainty. For instance, if it is known a priori that there is a large building in the line of sight of a beacon, if the beacon can no longer be seen, it suggests that the beacon lies behind the building, rather than between the camera and the building.

3.3 The Challenge of Exploiting Prior Information in SLAM

Consider again the situation shown in Figure 3.1: the robot is equipped with a sensor such as a laser range finder or a camera, and can pick up point features on building walls, and the satellite edges of

building walls. The types of maps built by high-speed, high-altitude UAVs equipped with a LIDAR sensor will be very different from those constructed by low-speed UGVs (unmanned ground vehicles) equipped with stereo cameras and laser range finders.

Using these maps in SLAM implies that we can express a mathematical relationship between their features, and determine whether this relationship holds. Traditional feature association methods, such as Joint Compatibility Branch and Bound (JCBB) [88] are not necessarily sufficient, as the maps are likely to have been generated by different sensing modalities from highly different positions, and thus will pick up different things in the environment. In addition they will be subject to missing and spurious features, and high position uncertainty. This results in a number of difficulties compared to standard data association.

The standard data association problem uses the same platform and sensor modality, so observations generated in the same part of the environment are likely to generate the same features, provided the feature extraction method extracts salient features and is able to reduce the incidence of spurious features. For instance if a sensor picks up railings in front of a building as salient features, on revisiting the area the sensor can be expected to once again pick up these same features (provided the environment remains relatively static). However a prior map will often have been generated by a different sensing system, which even for a homogeneous map representation would likely have picked up entirely different features. Thus features present in one of the maps will be absent in the other. If the map uncertainty is sufficiently high, features that correspond to different objects in the environment may appear to gate with each other. Such association failures are particularly difficult because they may only be detected once the platform closes a loop; if the error is large enough the platform may fail to recognise that it has closed the loop [39] too. This has motivated the use of non-geometric loop closure detection methods, such as FAB- MAP [24].

Because the sensor is mounted on the platform, the errors between the (projected) sensor features and SLAM state features are highly correlated through the platform pose (provided they were seen recently), and thus the association uncertainty mainly comes through sensor noise, which for accurate sensors such as SICK scanners is low. However because the prior map was generated independently of the SLAM state, they are uncorrelated and thus there is a greater degree of uncertainty when associating between them. This parallels the loop closure problem, where beacons not seen in a significant amount of time (and thus poorly correlated with the vehicle) are re-observed.

For these reasons for incremental association between features in the SLAM state and those in the prior map we place emphasis on being able to formulate a prior from non-geometric information, and use this with geometric information in a MAP-based framework. Thus we only commit to using prior information if the probability (rather than simply likelihood) of that information applying is high enough.

In addition, we want to be able to express feature characteristics from structure knowledge. For instance, if a feature arose from a car, it is likely to be dynamic; likewise buildings imply an urban area with buildings and roads nearby. Classifiers for labelling which objects (such as cars) a feature is part of have been developed [99, 76, 2] however they have yet to be exploited in this way.

Figure 3.2: Example showing a feature map whose map model G_w extracts straight edges on all structures in a 2D world; there are 5 such edges. In this case each surface may be described in the feature map $\mathbf{x}_w$ by a single line segment feature (shown as a black line), whose elements are the coordinates of the two end points.

In the following sections we formalise the above intuition, presenting a framework by which models expressing semantic knowledge of structures could be leveraged to relate features in one or more maps each other in a probabilistic way. We do not seek to create such models, rather we wish to be able to express them in such a way that they can be used to inform the SLAM process.

3.4 Modelling Prior Information

Consider again the situation shown in Figure 3.1. The robot has not visited the environment before, however prior information in the form of a map based on aerial imagery has become available. Although the features are dissimilar (*heterogeneous*) due to the different sensors and viewpoints, there is a common underlying *structure* generating these features - the building. Because of this common underlying cause, features in different feature maps are not independent of one another, and this makes it possible to pass information between them. Note that we assume that the errors, given association variables, are conditionally independent.

Consider Figure 3.2, which gives a simple example of a world consisting of 3 structures represented by grey shapes, from which line segment features have been extracted. The world is assumed to be populated by a set of structures, each of which is an *instance* of a *structure class*. The semantically meaningful analogy of a structure is an object such as a building, however our concept of structures is more general. The development of exhaustive taxonomies and object classes of world models are a major topic [82, 21, 86]; our intent is to have something sufficiently descriptive to support relationships between maps. Therefore, we only support part-of relationships: for example a window can be part of a wall, a wall may be part of a building, and a building may be part of a city. All structures of interest in the world can be considered to be an instance of one of the set of *structure classes* π, $\pi \in \{1 \dots n\}$. For the structure class $\mathbf{s}_n$, an oracle Λ can partition the world into the set of all instances of this structure, the ith instance being denoted $\mathbf{s}_{n\{i\}}$.

A sensing system typically consists of a physical measurement device together with signal processing, segmentation and detection algorithms. As a result, a particular sensing system induces certain *features* on an instance of a structure. In Figure 3.2 these are shown as line segments. The extracted features are defined by a feature map model G, which embodies the characteristics of the sensing system. Different sensing systems will in general induce different sets of features with different dimensionalities on different parts of the same structure. The features for all the sensing systems are maintained in a set

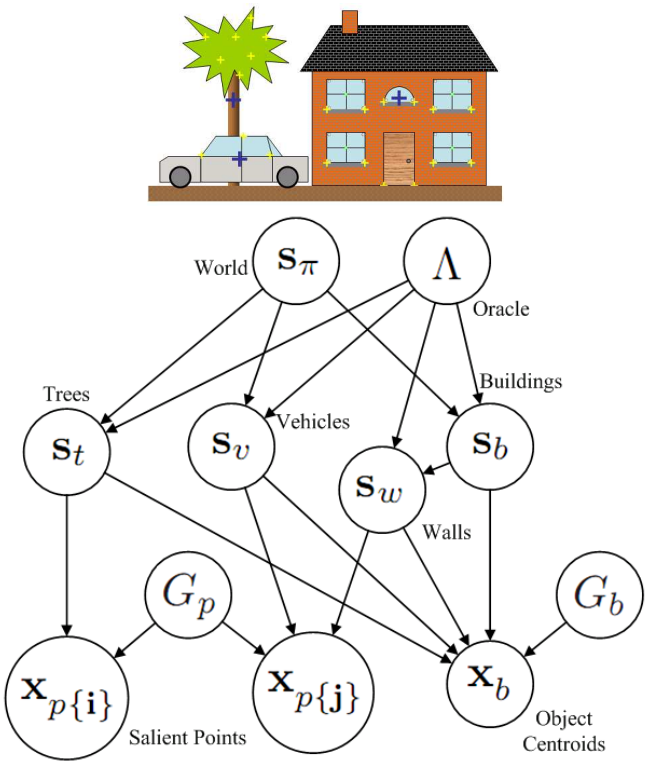

Figure 3.3: Top: Example of features that may be induced from structures. The large crosses represent the centroids of the building, car and tree, while the small crosses represent salient points detected by the LIDAR. Bottom: Example structure class hierarchy corresponding to the top picture, defined by the oracle Λ with the feature maps $\mathbf{x}_p$ and $\mathbf{x}_b$ being produced from them (under the feature map models G_p and G_b), along with labels of what they represent. In this case $\mathbf{s}_v$ and $\mathbf{s}_b$ are structure classes at the same level, and $\mathbf{s}_w$ is at a level lower as it is derived from $\mathbf{s}_b$. Feature maps may be produced from instances of structure classes at all levels of the hierarchy.

of *feature maps*, $m \in \{1 \dots M\}$ (or simply *maps*). Each feature map contains features of the same type. The mth feature map $\mathbf{x}_m$ is parameterised the same way, which describes the structures from which it is derived in terms of the sensing system. The features within it are stable, repeatedly observable aspects of an structure represented by a state, such as a set of points in space, texture patches or the centroid of a particular structure instance.

The pth feature map can be written as

$$\mathbf{x}_p = \left\{ \mathbf{x}_{p\{1\}}, \mathbf{x}_{p\{2\}}, \dots \mathbf{x}_{p\{N\}} \right\}, \tag{3.1}$$

where $\mathbf{x}_{p\{i\}}$ is the ith feature (or set of features) in the map. Thus in Figure 3.2, the map model G_w extracts line segments from the straight edges of the structures, and stores the coordinates of the end points of each line segment in $\mathbf{x}_w$.

We can generalise the simple example above to form a more complex network for several feature maps obtained from structure instances within an structure class hierarchy composed of several structure classes, as shown in Figure 3.3. In the example feature maps are produced from three structure classes

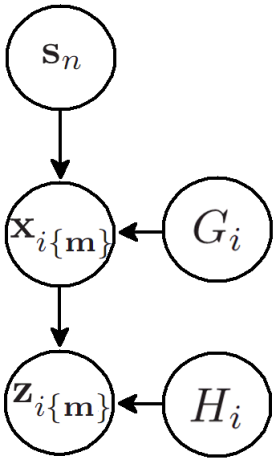

Figure 3.4: Bayes network showing the set of $\mathbf{m}$ features in the ith feature map $\mathbf{x}_{i\{\mathbf{m}\}}$, which have been derived from the structure class $\mathbf{s}_n$ under the feature map model G_i, from which a set of observations $\mathbf{z}_{i\{\mathbf{m}\}}$ have been made under the observation model H_i.

(buildings, trees and cars). Note that features do not have to be exclusive to a single feature map, so there may be overlapping features across maps.

A system makes observations of subsets of these features to create an estimate of a feature map.

3.4.1 Observations of Feature Maps

Observations are measurements $\mathbf{z}$ of features within a feature map, made under an observation model H and usually with respect to a local coordinate frame (for instance with respect to the pose of the platform)

$$\mathbf{z}_p(k) = \left\{\mathbf{z}_{p\{1\}}, \mathbf{z}_{p\{2\}}, \ldots \mathbf{z}_{p\{n\}}\right\}_k, \tag{3.2}$$

where at time k $\mathbf{z}_{p\{i\}}(k)$ is the measurement of the ith feature in the pth feature map, as shown in Figure 3.4. The measurements may not have the same form as the features in the feature map. For instance, a measurement of a 3D point feature by a camera will result in the projection of the feature into the camera frame, thus giving a 2D observation. Specifically $\mathbf{z}_p(k)$ is a function of $\mathbf{x}_p$ determined by the *observation model* $H_p(k)$. Unlike $\mathbf{x}_p$, $\mathbf{z}_p(k)$ is a function of the platform pose and observation noise,

$$\mathbf{z}_p(k) = \mathbf{h}(\mathbf{x}_v(k), \mathbf{x}_p, H_p(k), \mathbf{w}(k)), \tag{3.3}$$

where $\mathbf{x}_v(k)$ is the platform pose and $\mathbf{w}(k)$ is observation noise.

Maps are inferred from a set of observations $\mathbf{z} = \mathbf{z}(k_1 \ldots k_n)$ under a given map model G and observation model $H(k_1 \ldots k_n)$. The discrete data association parameter $\mathbf{c}(k_1 \ldots k_n)$ as commonly used in SLAM [43], which we consider to be part of the observation model, determines which map feature the observation goes with. Thus the full estimate of a feature map is $p(\mathbf{x}_p|H_p(k_1 \ldots k_n), G_p, \mathbf{z}_p(k_1 \ldots k_n))$, as shown in Figure 3.5. For brevity we make the models implicit, giving $p(\mathbf{x}_p|\mathbf{z}_p(k_1 \ldots k_n))$.

Having formally defined how we consider systems to produce a set of feature maps, and how there is shared information between them if they came from a common structure, we can now describe the

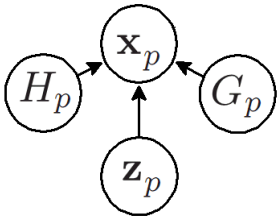

Figure 3.5: Bayes network describing a map $\mathbf{x}_p$ conditioned on a set of observations $\mathbf{z}_p$ under the observation model H_p and map model G_p.

form of this shared information, and perform inference as to whether this information applies.

3.4.2 Using structure information to inform features

Information of a usable form exists between features in maps if they were generated by a common known structure instance; this determines the form of the shared information between them. Thus for each feature we define a binary indicator function that expresses the statement "is the feature associated with (generated by) the structure $\mathbf{s}_{w\{s\}}$?",

$$d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\}) = \begin{cases} 1 & \text{if } \mathbf{s}_{w\{s\}} \text{ generated } \mathbf{x}_{p\{i\}} \\ 0 & \text{otherwise} \end{cases}. \tag{3.4}$$

If multiple features, such as a point i in feature map p and a line n in feature map l are associated with the structure $\mathbf{s}_{w\{s\}}$, this is indicated by

$$d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\}) \centerdot d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}), \tag{3.5}$$

where $x \centerdot y$ denotes logical conjunction (multiplication). This generalises to any number of feature maps, and indeed multiple features in one map can to be constrained by a single feature in another map. However for the purposes of illustration we will consider the case of two maps.

If $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ (which for brevity we can abreviate to $d_{in,s} = 1$) then this implies that

$$\mathbf{f}\left(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}\right) = \begin{cases} \ominus(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) & \text{for } d_{in,s} = 1 \\ \emptyset & \text{otherwise} \end{cases}, \tag{3.6}$$

where $\mathbf{f}(\cdot)$ is a non-trivial function with a unique value[1] and $\ominus(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$ is the parameterisation of the relationship; thus

$$\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = \theta_{in}, \tag{3.7}$$

where θ_{in} is a unique parameter encoding a degree of freedom. The value of θ_{in} depends on the feature types, and it has a prior $p(\theta_{in}|d_{in,s} = 1)$ conditioned on the structure class w and instance (or set of

[1] Although this is not a requirement we make this assumption here for simplicity

instances) s. $p(\theta_{in}|d_{in,s}=1)$ thus represents the distribution of θ across all features in $\mathbf{x}_p$ and $\mathbf{x}_m$ that are associated with $\mathbf{s}_{w\{s\}}$. For instance, if θ is the distance of the ith point $\mathbf{x}_{p\{i\}}$ from the line segment $\mathbf{x}_{m\{n\}}$, where both are part of the wall structure $\mathbf{s}_{w\{s\}}$, and it is known that all points on this wall are uniformly distributed within a certain region, $p(\theta_{in}|d_{in,s}=1)$ would be uniform over that range.

We can perform inference using this model as follows. For brevity let us denote the set of maps $\mathbf{x}_{pm}=\{\mathbf{x}_p,\mathbf{x}_m\}$ and associated observations $\mathbf{z}_{pm}=\{\mathbf{z}_p(k_1\dots k_a),\mathbf{z}_m(k_1\dots k_b)\}$. If we know that $d_{in,s}=1$, then the posterior probability of the maps conditioned on the observations $\mathbf{z}$ is given by Bayes rule,

$$p(\mathbf{x}_{pm}|d_{in,s}=1,\mathbf{z}_{pm})=\eta\, p(d_{in,s}=1|\mathbf{x}_{pm})\, p(\mathbf{x}_{pm}|\mathbf{z}_{pm}), \tag{3.8}$$

where η is a normalising term.

The concept of the description of the world presented above is that any map generated by a system is a feature map. The presence of shared information between feature maps p and l is expressed by an underlying structure $w\{s\}$ between them, and is indicated by $d(\mathbf{x}_p,\mathbf{x}_l,\{\mathbf{s}_{w\{s\}}\})$. For now we will assume that the value of this indicator is known, that is we know that the common underlying structure between the feature maps applies, however later in this chapter we will show how we can infer its value.

Consider an agent performing SLAM. If there is shared information between the map in the SLAM state and a prior map, the prior map can be used to improve the SLAM map and thus the pose estimate of the agent. As a prior map represents an external reference, it is independent of the map model and observation model of the agent and its sequence of observations, and thus should give significant benefits to SLAM. The prior map may mitigate modelling, representational and linearisation errors in the SLAM system, such as those present in EKF-SLAM [61].

To determine whether a prior map does indeed significantly improve SLAM, we now discuss how we can implement the framework for EKF-SLAM when the underlying structure is known, and perform experiments with various map accuracies. We start by presenting the form of the prior map we use throughout the thesis.

3.5 SLAM with a Line Segment Prior Map

The prior information considered in this thesis is assumed to consist of a geometric map; that is we have a map consisting of line segments or points whose position in the real world are known to a certain degree of accuracy. In general the accuracy of the prior map is higher than that of the map produced by an unaided agent performing SLAM (and thus whose error grows without bound). The features on the prior map would correspond to landmarks in the map, such as building walls. This map is assumed to have been extracted a priori from aerial data (as depicted in Figure 3.6), LIDAR data or otherwise. We wish to integrate this known data into our metric SLAM map (which consists of points), thus increasing its accuracy and the accuracy of the platform's position estimate as it moves around the environment.

The prior map for an EKF-based SLAM system, as in this thesis estimates n features in the feature map $\mathbf{x}_m$ by their mean $\hat{\mathbf{x}}_m$ and covariance $\mathbf{P}_m$. We are interested in the case where the SLAM map

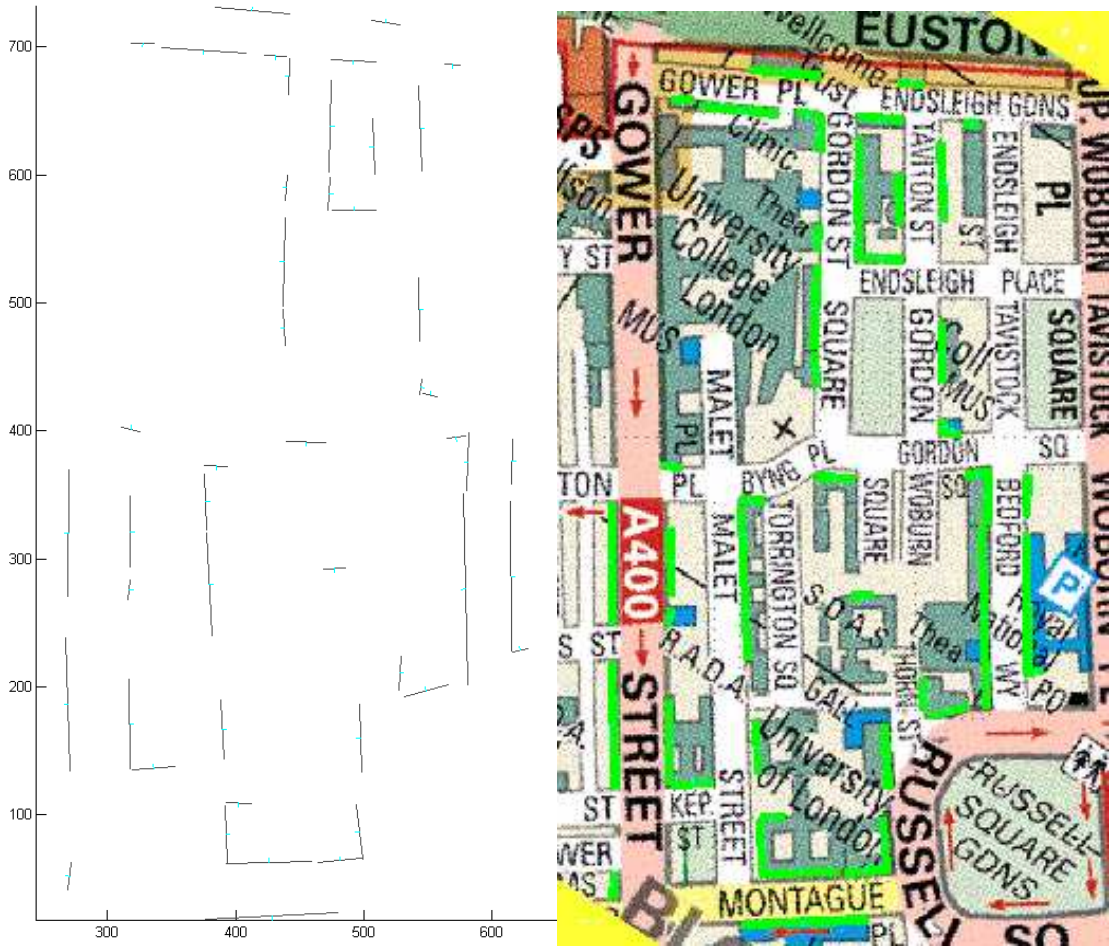

Figure 3.6: Left: line segments in the simulated environment introduced in chapter 2, representing flat vertical building walls near the platform trajectory. Right: the line segments overlayed on the map they were manually derived from, representing a part of London south-east of the UCL campus. Units are in m.

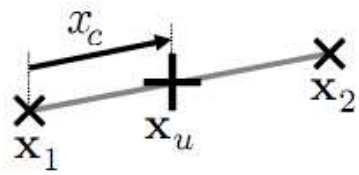

Figure 3.7: The point $\mathbf{x}_u$ may be represented in terms of the displacement x_c along the line segment defined by $\mathbf{x}_1$ and $\mathbf{x}_2$.

consists of 2D point features and the prior map consists of a series of separate line segments representing building walls in planar 2D space. The nth prior map feature has mean

$$\hat{\mathbf{x}}_{m\{n\}} = \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix}_{m\{n\}} \tag{3.9}$$

and covariance

$$\mathbf{P}_{m\{n\}} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12}^T \\ \mathbf{P}_{12} & \mathbf{P}_{22} \end{bmatrix}_{m\{n\}}. \tag{3.10}$$

In our implementation $\mathbf{x}_{p\{i\}}$ is a point which lies on the line segment $\mathbf{x}_{m\{n\}}$ that passes through the points $\mathbf{x}^1_{l\{n\}}$ and $\mathbf{x}^2_{l\{n\}}$ if both are generated by the same flat wall structure instance $\mathbf{s}_{w\{s\}}$, $\mathbf{x}_{p\{i\}}$ and $\mathbf{x}_{m\{n\}}$ being the true feature positions. Then $\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$ is the normal distance between the point and line segment,

$$\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = \frac{\det\left(\begin{bmatrix} \mathbf{x}^2_{m\{n\}} - \mathbf{x}^1_{m\{n\}} & \mathbf{x}^1_{m\{n\}} - \mathbf{x}_{p\{i\}} \end{bmatrix}\right)}{\left|\mathbf{x}^2_{m\{n\}} - \mathbf{x}^1_{m\{n\}}\right|}, \tag{3.11}$$

and $\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = \theta_{in} = 0$ because the point lies on the line segment if $d_{in,s} = 1$. Thus $p(\theta_{in}|d_{in,s} = 1)$ is a delta function in our case. This means that the feature $\mathbf{x}_{p\{i\}}$ that lies on the line segment $\mathbf{x}_{m\{n\}}$ can be represented by the relation

$$\mathbf{x}_{p\{i\}} = \mathbf{f}(x_{c\{i\}}, \mathbf{x}_{m\{n\}}) = \mathbf{x}^1_{m\{n\}} + x_{c\{i\}}\left(\mathbf{x}^2_{m\{n\}} - \mathbf{x}^1_{m\{n\}}\right), \tag{3.12}$$

where $x_{c\{i\}}$ is the distance of the point along the line existing between the points $\mathbf{x}^1_{m\{n\}}$ and $\mathbf{x}^2_{m\{n\}}$, as shown in Figure 3.7.

The line segments representing flat, vertical building walls in the environment described in chapter 2 and shown in Figure 3.6 can be considered a feature map $\mathbf{x}_m$. A noisy prior map representing all the 48 walls in the map is made available at the start of each run. The covariance of the map in m^2 is

$$\mathbf{P}_m = \text{diag}\left(P_m\right), \tag{3.13}$$

where P_m is the variance of each element, all elements having the same variance. The prior map mean is

$$\hat{\mathbf{x}}_m = \mathbf{x}_m + \aleph(\mathbf{0}, \mathbf{P}_m), \tag{3.14}$$

where $\aleph(\mu, \Sigma)$ is Gaussian distributed noise with mean μ and covariance Σ, $\hat{\mathbf{x}}_m$ computed at the start of each run. $\hat{\mathbf{x}}_m$ has dimension 192 (there are 4 elements per line segment). The platform is assumed not to be in communication with any other agents or ground stations, and thus only has access to this version of the prior map during the run.

3.6 EKF-SLAM Implementation

Given $d_{in,s} = 1$, we can evaluate (3.50) as follows. At time k $p(\mathbf{x}_p|\mathbf{z}_p(1:k))$ and $p(\mathbf{x}_m|\mathbf{z}_m)$ are the map estimates from the SLAM state and prior map respectively; here the observations $\mathbf{z}_p(1:k)$ and $\mathbf{z}_m$ are assumed independent however this condition is not required.

When a platform performing SLAM receives a prior map, it is appended to the SLAM state from (2.9) and (2.10) as

$$\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_p^T & \hat{\mathbf{x}}_m^T \end{bmatrix}_{k|k}^T, \tag{3.15}$$

$$\mathbf{P}(k|k) = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{vp}^T & 0 \\ \mathbf{P}_{vp} & \mathbf{P}_p & 0 \\ 0 & 0 & \mathbf{P}_m \end{bmatrix}_{k|k}, \tag{3.16}$$

representing the mean and covariance of the joint vehicle/map/prior map estimate $p(\mathbf{x}_v(k), \mathbf{x}_p, \mathbf{x}_m|\mathbf{z}_m, \mathbf{z}_p(1:k))$.

The prior map information is applied to beacons as follows. When a new beacon is seen it is initialised into the state as in regular SLAM. A method such as Euclidean gating can be used to determine which features in the prior map have a common structure k with beacons in the state. In this thesis we use the scenario where point beacons in the state may relate with prior map line segments representing walls. We assume here that we have already established that $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ for the ith point feature in the state and nth line segment feature in the prior map (thus (3.12) holds), however in the next chapters we will show how $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\})$ can be inferred from the observations.

Our task is to condition our estimate on this structure, thus to estimate the posterior map and platform estimates $p(\mathbf{x}_v(k), \mathbf{x}_{pm}|\mathbf{z}_{pm}, d(\mathbf{x}_p, \mathbf{x}_m, \{\mathbf{s}_{w\{s\}}\}) = 1)$. Before the structure has been exploited, the estimate of the two maps is given by

$$p(\mathbf{x}_p, \mathbf{x}_m|\mathbf{z}_p(1:k), \mathbf{z}_m) = p(\mathbf{x}_p|\mathbf{z}_p(1:k))\, p(\mathbf{x}_m|\mathbf{z}_m), \tag{3.17}$$

as the two are independent.

The relation implies that (3.12) holds, and thus we can represent the *unconstrained* form $\hat{\mathbf{x}}_{p\{i\}}$ by the *constrained* form $\mathrm{f}(\hat{\mathbf{x}}_{m\{n\}}, \hat{x}_{c\{i\}})$. Thus instead of maintaining both $\hat{\mathbf{x}}_{p\{i\}}$ and $\hat{\mathbf{x}}_{m\{n\}}$ in the state, we estimate $\hat{\mathbf{x}}_{m\{n\}}$ and the extra parameter $\hat{x}_{c\{i\}}$.

We now need to initialise $\hat{x}_{c\{i\}}$ and $P_{c\{i\}}$ into the state. Initially it is added to the state with an uninformative (highly uncertain) variance which will be refined below. The intermediate state and covariance estimates (assuming this is the first time the prior map is used, and is thus uncorrelated with the rest of the SLAM state) for this first beacon are

$$\hat{\mathbf{x}}(k|k) = \left[\begin{array}{cccc} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_p^T & \hat{\mathbf{x}}_c^T & \hat{\mathbf{x}}_m^T \end{array} \right]_{k|k}^T, \tag{3.18}$$

$$\mathbf{P}(k|k) = \left[\begin{array}{cccc} \mathbf{P}_v & \mathbf{P}_{vp}^T & 0 & 0 \\ \mathbf{P}_{vp} & \mathbf{P}_p & 0 & 0 \\ 0 & 0 & \mathbf{P}_c & 0 \\ 0 & 0 & 0 & \mathbf{P}_m \end{array} \right]_{k|k}. \tag{3.19}$$

We use the information from $\hat{\mathbf{x}}_p$ to estimate an intial value for $\hat{\mathbf{x}}_c$ via a pseudo-observation [112]. The pseudo-observation specifies that the innovation between the constrained and unconstrained estimates is zero,

$$\nu(k) = \hat{\mathbf{x}}_{p\{i\}}(k|k) - \mathbf{f}(\hat{\mathbf{x}}_{m\{n\}}(k|k), \hat{x}_{c\{i\}}) = \mathbf{0}. \tag{3.20}$$

The covariance $\mathbf{S}$ of the innovation is

$$\begin{aligned} \mathsf{P}^{in} &= \nabla \mathbf{f}_i \left[\begin{array}{cc} \mathbf{P}_c^i & \mathbf{P}_{cm}^{in} \\ \left(\mathbf{P}_{cm}^{in}\right)^T & \mathbf{P}_m^n \end{array} \right]_{k|k} \nabla^T \mathbf{f}_i \\ \wp_{pc}^{in} &= \nabla \mathbf{f}_i \left[\begin{array}{cc} \left(\mathbf{P}_{pc}^i\right)^T & \left(\mathbf{P}_{pm}^{in}\right)^T \end{array} \right]_{k|k}^T \\ \mathbf{S} &= \left[\begin{array}{cc} \mathbf{P}_p^i & \wp_{uc}^{in} \\ \left(\wp_{pc}^{in}\right)^T & \mathsf{P}^{in} \end{array} \right]_{k|k}, \end{aligned} \tag{3.21}$$

where $\nabla \mathbf{f}_i$ is the Jacobian of $\mathbf{f}(\cdot)$ evaluated at $\hat{\mathbf{x}}(k|k)$, under the hypothesis that $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ (k representing the structure that gives rise to (3.12)).

The updated mean and covariance have the form

$$\hat{\mathbf{x}}(k|k) = \left[\begin{array}{cccc} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_p^T & \hat{\mathbf{x}}_c^T & \hat{\mathbf{x}}_m^T \end{array} \right]_{k|k}^T, \tag{3.22}$$

$$\mathbf{P}(k|k) = \left[\begin{array}{cccc} \mathbf{P}_v & \mathbf{P}_{vp}^T & \mathbf{P}_{vc}^T & \mathbf{P}_{vm}^T \\ \mathbf{P}_{vp} & \mathbf{P}_p & \mathbf{P}_{pc}^T & \mathbf{P}_{pm}^T \\ \mathbf{P}_{vc} & \mathbf{P}_{pc} & \mathbf{P}_c & \mathbf{P}_{cm}^T \\ \mathbf{P}_{vm} & \mathbf{P}_{pm} & \mathbf{P}_{cm} & \mathbf{P}_m \end{array} \right]_{k|k}. \tag{3.23}$$

At this point having initialised $\hat{x}_{c\{i\}}$, $\hat{\mathbf{x}}_{p\{i\}}$ is redundant so we remove it and maintain only the constrained form $\mathbf{f}(\hat{\mathbf{x}}_{m\{n\}}, \hat{x}_{c\{i\}})$, estimating $\hat{\mathbf{x}}_{m\{n\}}$ and $\hat{x}_{c\{i\}}$ jointly with the vehicle and rest of the map as in regular SLAM.

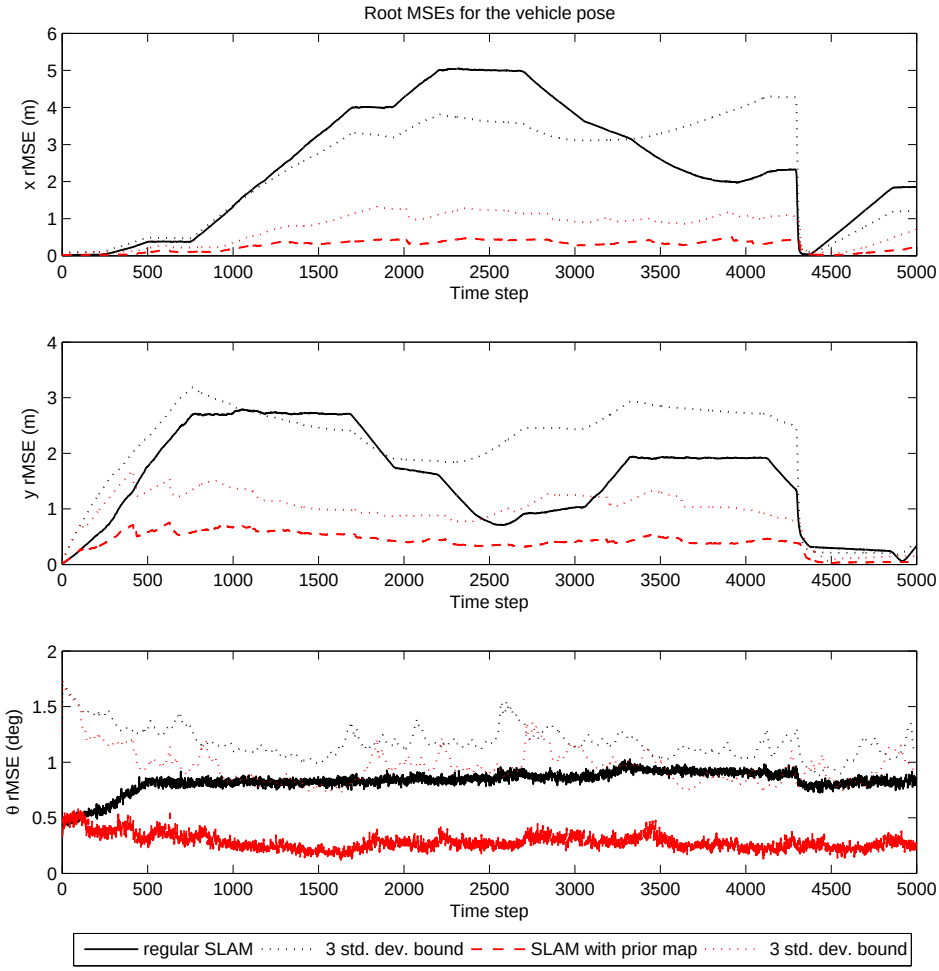

Figure 3.8: Comparison of the vehicle pose error for regular Second Order EKF-SLAM with and without a prior map. The map has a 1 m std. dev. error. The estimate with a prior map is more accurate, and remains within its 3σ estimate bounds, showing the accuracy and consistency gains possible when using prior information.

3.6.1 Initial Results

Let us now show the effect of introducing a prior map on SLAM in the simulation environment described in chapter 2. The results presented are averaged over 100 Monte Carlo runs.

Figure 3.8 shows that for SLAM with an accurate (0.01 m 1σ range error) sensor and inaccurate (1 m 1σ error) prior map, the vehicle pose estimate is more accurate and consistency is improved compared to regular SLAM. In addition the growth in uncertainty is limited. This occurs because as the platform moves around the environment, new line segments are added to the state, and their information propagates to the SLAM map and vehicle estimate, reducing their uncertainty. In our case, the prior map is consistent, so when combined with the inconsistent EKF-SLAM estimate it acts as a "buffer", reducing the degree of inconsistency in the map and vehicle pose (however in doing so its estimate becomes inconsistent). We use the Second Order filter to reduce the effect of linearisation errors.

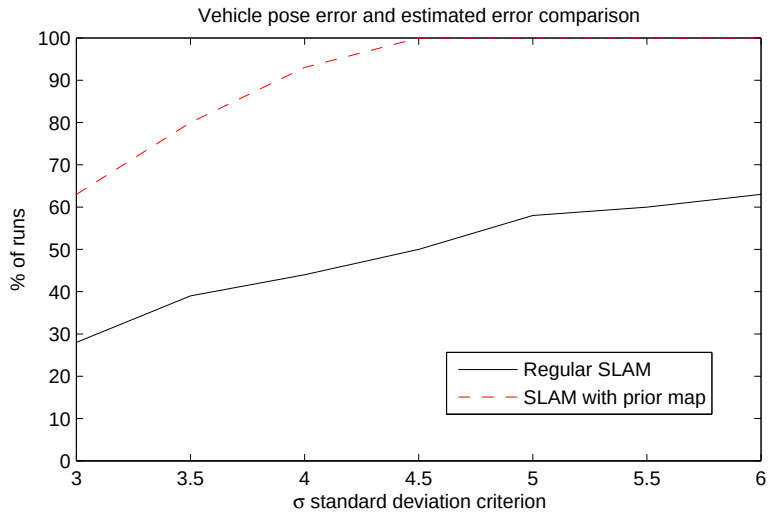

Figure 3.9: Consistency metric comparison between SLAM with and without a prior map.

Figure 3.9 shows the consistency metric, which indicates for what percentage of runs the vehicle pose was within a given standard deviation criterion for at least 95% of the run. Significantly more runs met the consistency criterion for SLAM with a prior map than without. At the 3σ criterion twice the number of runs were within this bound for SLAM with a prior map, and all the runs were within the 4.5σ criterion, compared to only 50% of the runs for SLAM without a prior map. This confirms that the prior map has significantly improved the consistency of SLAM.

However, despite the significant error decrease and consistency improvement, under certain conditions the estimate can become corrupted and thus the above solution is inadequate on its own, as we will now show. Figure 3.10 shows an example of the effect of using a prior map with a 2 m std. dev. error in SLAM. We have increased the vehicle steer error standard deviation from 3 to 6° to clearly show the error. For a relatively accurate map (1 m std. dev.), even at 6° steer error there is no obvious corruption - the state appears consistent. However for the more uncertain (2 m) map, the state has become corrupted resulting in the greater vehicle pose error and inconsistency.

Corruption also occurs for the vehicle with 3° std. dev. steer error when we reduce the sensor bearing uncertainty from 1.1° to 0.1° (1σ error). Figure 3.11 compares regular Second Order EKF SLAM (without a prior map) for the two sensor noises, the results being averaged over 100 MC runs. Note how the orientation error sharply increases between 300 and 500 time steps, indicating rotational "map slip" [106] (the map estimate rotates relative to the ground truth). There is no obvious reason for why this consistently happens there.

When we introduce a 1 m std. dev. prior map the severity of the error is greatly reduced, however it is still present as shown by the inconsistent vehicle pose in Figure 3.12.

As discussed in chapter 2, these errors occur because EKF-based SLAM suffers from corruption of its estimate due to linearisation errors and a fundamental deficiency in the way the EKF handles orientation uncertainty [9, 58, 61]. The prior information reduces the uncertainty in the system and can qualitatively mitigate the EKF and linearisation errors that cause this corruption, however its effective-

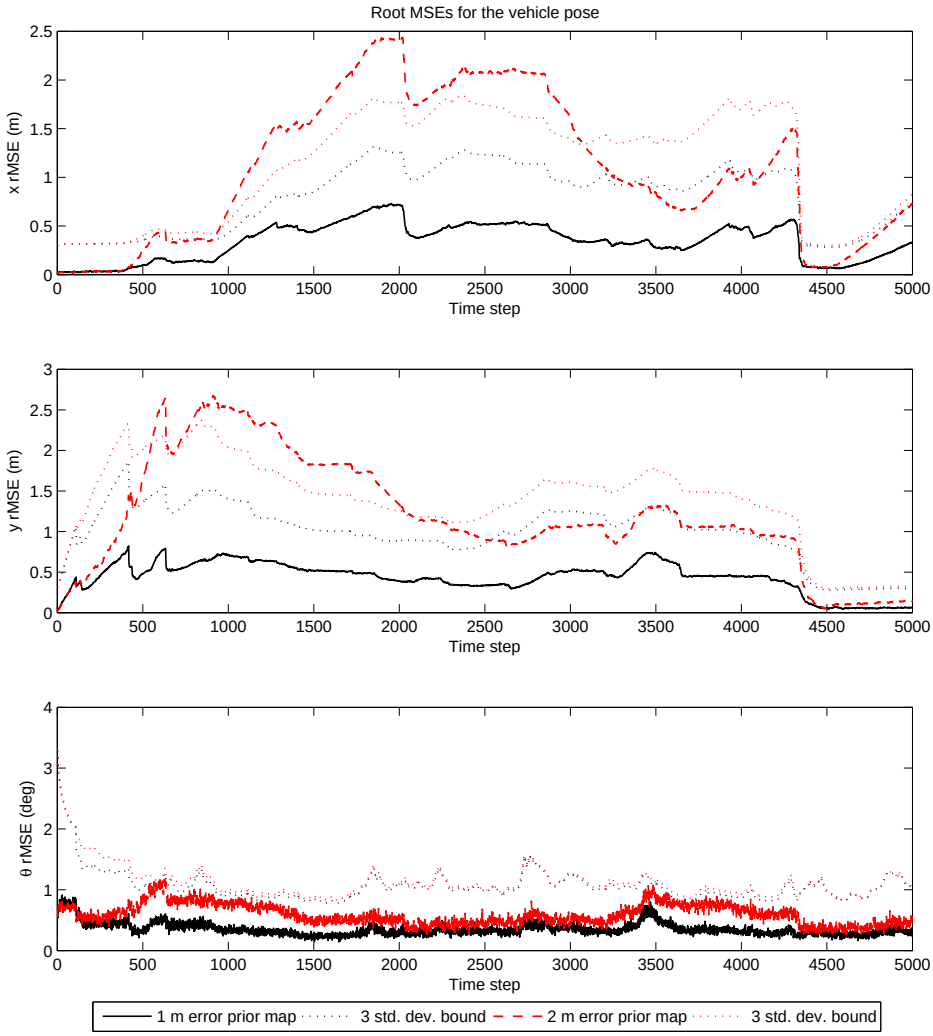

Figure 3.10: Increased error and inconsistency of the vehicle pose for a vehicle with 6° std. dev. steer noise, when the prior map error is increased from 1 m to 2 m std. dev. The weaker prior map information makes the SLAM system more vulnerable to corruption (caused by EKF and linearisation errors).

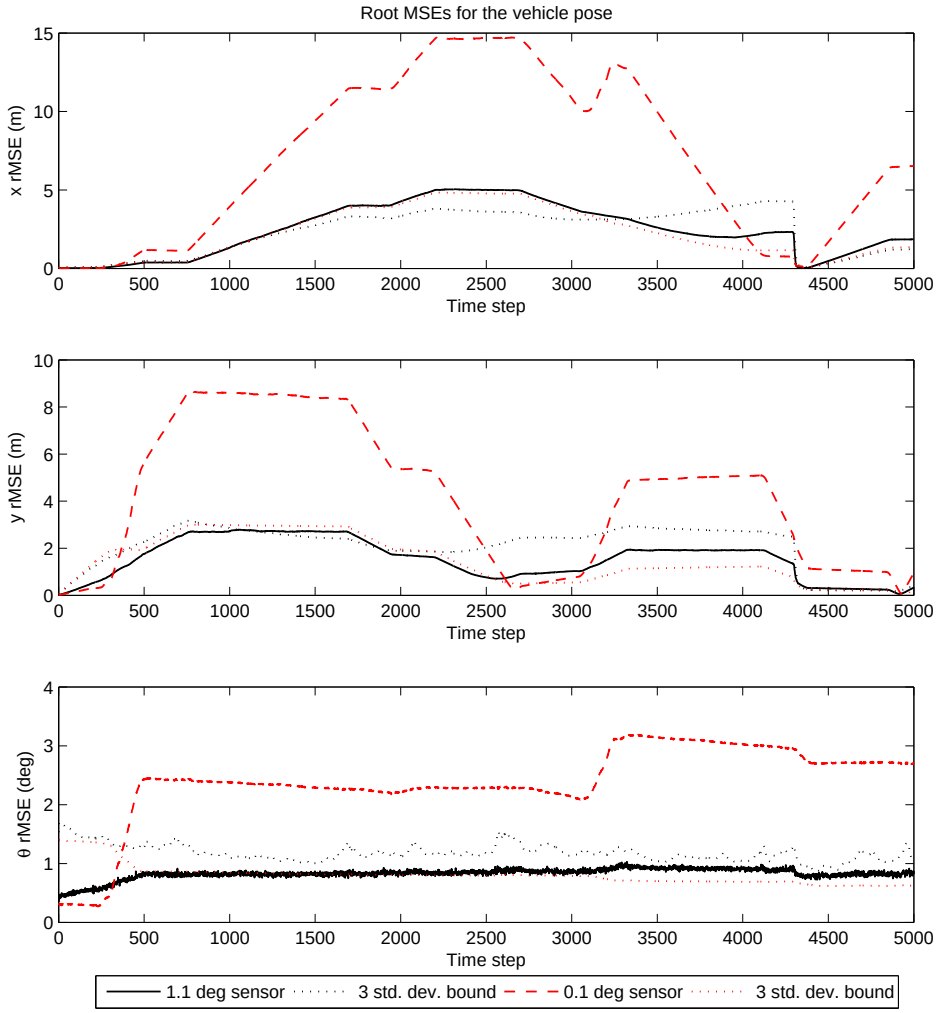

Figure 3.11: Changes in model parameters can have non-obvious effects on EKF errors. This comparison of the vehicle pose error between Second Order filter SLAM runs with two sensor bearing accuracy levels ($1.1°$and $0.1°$) shows how map slip suddenly obviously manifests itself for the more accurate sensor.

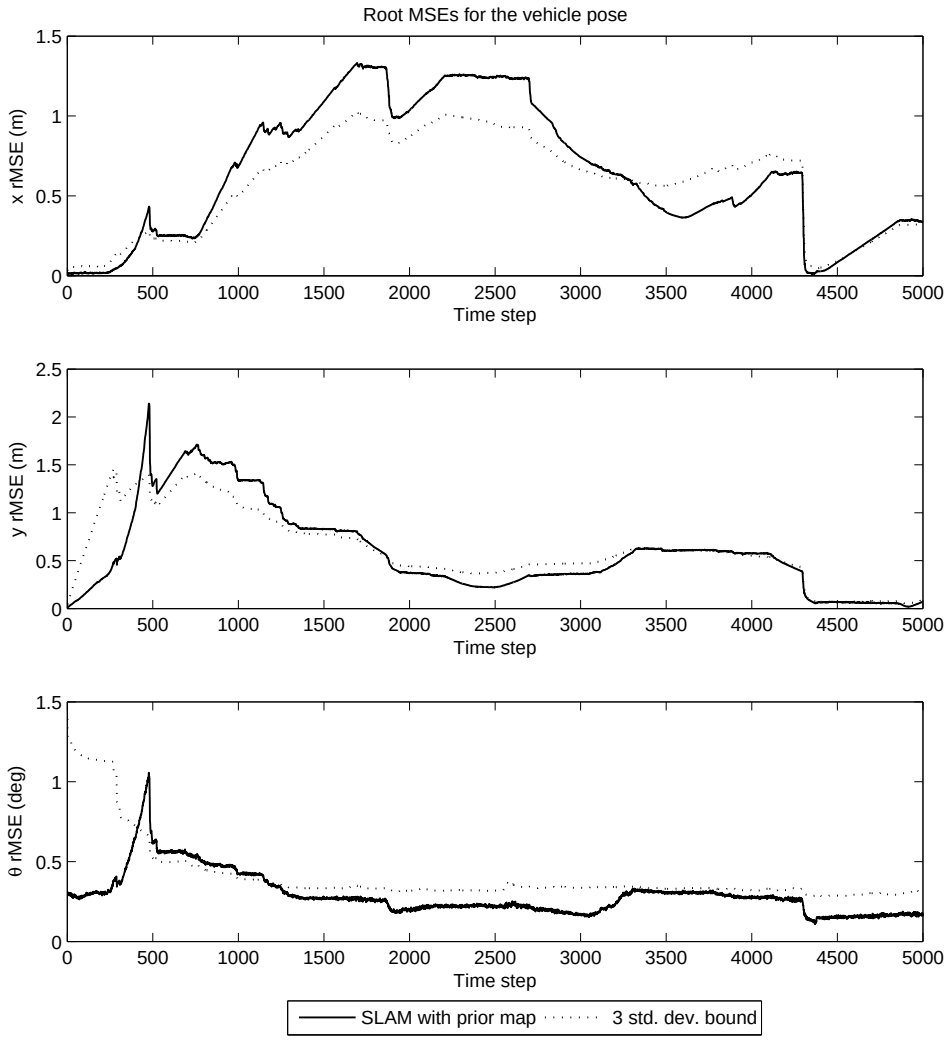

Figure 3.12: Effect of using a prior map with the SLAM system in Figure 3.11, whose sensor bearing 1σ error is $0.1°$. The vehicle pose recovers from the map slip at 500 time steps, however there is still noticeable inconsistency, as the error should be far lower than the 3σ uncertainty estimate shown.

ness at doing so diminishes as the uncertainty in the prior map increases. These errors will manifest themselves to a greater or lesser extent depending on the models and noises in the system, so may not be immediately noticeable [61].

To reduce the severity of these effects, and other sources of corruption such as modelling errors we introduce a novel parameterisation for beacons with prior information, which we call the *Dual Representation* [95].

3.6.2 The Dual Representation

The Dual Representation allows us to use prior information to improve the SLAM process in terms of both reducing uncertainty, reducing corruption in the state and rendering the prior information itself immune to corruption resulting from the SLAM process. With the Dual Representation we keep the two constrained and unconstrained beacon representations in the state, which has the form of (3.22) and (3.23) above. The constrained estimate $\mathbf{f}(\hat{\mathbf{x}}_{m\{n\}}, \hat{x}_{c\{i\}})$ maintained by the Dual Representation differs from that of the regular constrained form in that the prior map estimate $\hat{\mathbf{x}}_{m\{n\}}$ is never updated. This is why for an optimal estimate the Dual Representation also maintains the unconstrained estimate $\hat{\mathbf{x}}_{p\{i\}}$. The redundancy in the Dual Representation using more elements than the regular constrained (line segment) form poses no problem in theory, provided we account for the correlations [60].

3.6.2.1 Updating Beacons in Dual Representation Form

When new observations are made, we estimate both $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_c$ as follows. The observations received by the robot at time k are contained in the vector $\mathbf{z}_u(k)$. A new observation vector is formed by duplicating the observations to form

$$\mathbf{z}(k) = \begin{bmatrix} \mathbf{z}_p \\ \mathbf{z}_m \end{bmatrix}_k, \mathbf{z}_p = \mathbf{z}_m \tag{3.24}$$

where $\mathbf{z}_p$ and $\mathbf{z}_m$ are observations of unconstrained and constrained beacons respectively.

These relate to the state through the observation functions

$$\mathbf{z}_p = \mathrm{h}(\mathbf{x}_p, \mathbf{w}) \tag{3.25}$$

and

$$\mathbf{z}_m = \mathrm{h}(\mathbf{f}(\mathbf{x}_m, \mathbf{x}_c), \mathbf{w}) = \mathrm{g}(\mathbf{x}_m, \mathbf{x}_c, \mathbf{w}), \tag{3.26}$$

where $\mathbf{w}$ is zero-mean noise with covariance $\mathbf{R}$, $\mathbf{x}_m$ is the prior map and the p and m subscripts refer to unconstrained and constrained beacons respectively. In our case, that of planar range-bearing SLAM (3.25) is given by

$$\mathrm{h}(\mathbf{x}_v, \mathbf{x}_p, \mathbf{w}) = \begin{bmatrix} z_r \\ z_\phi \end{bmatrix}_u = \begin{bmatrix} \sqrt{(x_p - x_v)^2 + (y_p - y_v)^2} + w_r \\ \arctan(\frac{(y_p - y_v)}{(x_p - x_v)}) - \theta_v + w_\phi \end{bmatrix}_k, \tag{3.27}$$

where $\mathbf{x}_v = \begin{bmatrix} x & y & \theta \end{bmatrix}_v^T$ is the vehicle pose, $\mathbf{x}_p = \begin{bmatrix} x_p & y_p \end{bmatrix}^T$ is the unconstrained beacon and $\mathbf{w} = \begin{bmatrix} w_r & w_\phi \end{bmatrix}^T$ is noise, and (3.26) by

$$\mathrm{g}(\mathbf{x}_v, \mathbf{x}_m, \mathbf{x}_c, \mathbf{w}) = \begin{bmatrix} z_r \\ z_\phi \end{bmatrix}_c = \begin{bmatrix} \sqrt{(x_m + x_c(x_2 - x_1)_m - x_v)^2 + (y_m + x_c(y_2 - y_1)_m - y_v)^2} + w_r \\ \arctan(\frac{(x_u^2 - y_v)}{(x_u^1 - x_v)}) - \theta_v + w_\phi \end{bmatrix}, \quad (3.28)$$

where $\mathbf{x}_m = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 \end{bmatrix}_m^T$ is the line segment and x_c is the distance along the line segment.

Beacons that do not have relations, and are thus only represented by $\mathbf{x}_p$ are updated solely using (3.25) as in regular SLAM [23]. For beacons that do have two representations in the state, both representations are updated together. Thus the observation noise covariance is

$$\mathbf{R}(k) = \begin{bmatrix} \mathbf{R}_p & \mathbf{R}_p^T \\ \mathbf{R}_p & \mathbf{R}_m \end{bmatrix}_k, \mathbf{R}_p = \mathbf{R}_m. \quad (3.29)$$

The observation Jacobians of (3.25) and (3.26) are $\nabla\mathbf{h}$ and $\nabla\mathbf{g}$ respectively. $\nabla\mathbf{g}$ has the form

$$\nabla\mathbf{g}(k) = \begin{bmatrix} \nabla\mathbf{g}_v & \mathbf{0} & \nabla\mathbf{g}_p & \nabla\mathbf{g}_m \end{bmatrix}_k, \quad (3.30)$$

corresponding to (3.22), where $\nabla\mathbf{g}_v$ are the vehicle terms, $\nabla\mathbf{g}_p$ corresponds to beacons in constrained form and $\nabla\mathbf{g}_m$ are the prior map terms. The terms corresponding to $\hat{\mathbf{x}}_p$ are zero. $\nabla\mathbf{h}$ has the form

$$\nabla\mathbf{h}(k) = \begin{bmatrix} \nabla\mathbf{h}_v & \nabla\mathbf{h}_p & \mathbf{0} & \mathbf{0} \end{bmatrix}_k, \quad (3.31)$$

where $\nabla\mathbf{h}_v$ are the vehicle terms and $\nabla\mathbf{h}_p$ corresponds to the beacons in unconstrained form. The terms corresponding to $\hat{\mathbf{x}}_c$ and $\hat{\mathbf{x}}_m$ in (3.22) are zero.

Thus the update Jacobian $\nabla\mathbf{H}(k)$ has the form

$$\nabla\mathbf{H}(k) = \begin{bmatrix} \nabla\mathbf{h}_v & \nabla\mathbf{h}_p & \mathbf{0} & \mathbf{0} \\ \nabla\mathbf{g}_v & \mathbf{0} & \nabla\mathbf{g}_c & \nabla\mathbf{g}_m \end{bmatrix}_k, \quad (3.32)$$

the rows of $\nabla\mathbf{H}(k)$ corresponding to the observations in (3.24), and the columns corresponding to the individual components of the state in (3.33).

A Schmidt-Kalman update [103] is used to update all but the prior map state (the curly braces indicating which terms are not updated),

$$\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_p^T & \hat{\mathbf{x}}_c^T & \{\hat{\mathbf{x}}_m^T\} \end{bmatrix}_{k|k}^T, \quad (3.33)$$

$$\mathbf{P}(k|k) = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{vp}^T & \mathbf{P}_{vc}^T & \mathbf{P}_{vm}^T \\ \mathbf{P}_{vp} & \mathbf{P}_p & \mathbf{P}_{pc}^T & \mathbf{P}_{pm}^T \\ \mathbf{P}_{vc} & \mathbf{P}_{pc} & \mathbf{P}_c & \mathbf{P}_{cm}^T \\ \mathbf{P}_{vm} & \mathbf{P}_{pm} & \mathbf{P}_{cm} & \{\mathbf{P}_m\} \end{bmatrix}_{k|k}. \quad (3.34)$$

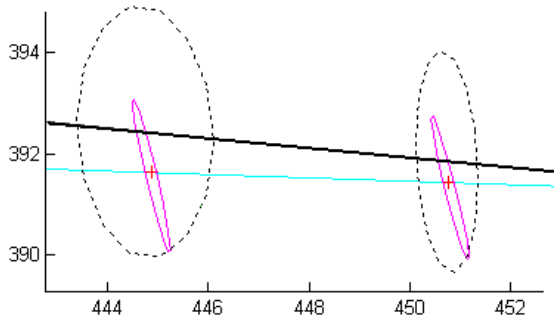

Figure 3.13: Line segment showing the constrained (dotted) and unconstrained (solid) 3σ ellipses for two beacons (crosses) lying on the wall (thin line). The accuracy and consistency of the unconstrained estimate is a direct result of the presence of the constrained estimates in the state. Because the prior map is never updated, the constrained estimates still lie on the prior map estimate of the wall (thick line).

The Schmidt-Kalman update equations are the same as in the standard Kalman update [23] with the exception of the addition of an indicator matrix with the form

$$\mathbf{W}(k) = \text{diag}\left(\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \end{bmatrix}\right), \tag{3.35}$$

with appropriately sized **1**s and **0**s to correspond to (3.33), where diag$(\cdot)$ is a diagonal matrix. For the Second Order filter described in Section 2.8 of chapter 2 the update equations are

$$\begin{aligned}
\nu(k) &= \mathbf{z}(k) - \begin{bmatrix} \mathrm{h}(\hat{\mathbf{x}}_p, \mathbf{0})^T & \mathrm{g}(\hat{\mathbf{x}}_m, \hat{\mathbf{x}}_c, \mathbf{0})^T \end{bmatrix}^T, && (3.36)\\
\mathbf{S}(k) &= \left[\nabla\mathbf{H}(k)\,\mathbf{P}(k|k-1)\,\nabla\mathbf{H}^T(k) + \mathbf{R} + \frac{1}{2}(\mathbf{M}(k)\,\mathbf{P}^2\,\mathbf{M}(k))\right]_k, && (3.37)\\
\mathbf{K}(k) &= \mathbf{W}(k)\,\mathbf{P}(k|k-1)\nabla\mathbf{H}^T(k)\,\mathbf{S}(k)^{-1}, && (3.38)\\
\hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k) \times [\nu(k) - \frac{1}{2}(\mathbf{P}(k|k-1)\,\mathbf{M}(k))], && (3.39)\\
\mathbf{J}(k) &= \mathbf{I} - \mathbf{K}(k)\,\nabla\mathbf{H}(k), && (3.40)\\
\mathbf{P}(k|k) &= \mathbf{J}(k)\,\mathbf{P}(k|k-1)\,\mathbf{J}(k)^T + \mathbf{K}(k)\,\mathbf{R}(k)\,\mathbf{K}(k)^T. && (3.41)
\end{aligned}$$

3.6.2.2 Comparison with the Line Segment Parameterisation

Figure 3.13 shows an example of the unconstrained and constrained estimates of two beacons using the Dual Representation with a line segment feature from the prior map. Note how the ellipses corresponding to the constrained estimates are suboptimal because they depend on the fixed prior map $\hat{\mathbf{x}}_m$.

Figure 3.14 shows a comparison between the vehicle pose error from the dual representation and that of the regular constrained parameterisation used in Figure 3.10, for the 2 m std. dev. prior map. The Dual Representation parameterisation has reduced the degree of corruption in the vehicle pose. This is particularly noticeable in the x and y position.

Figure 3.15 shows that the Dual Representation also reduces the inconsistency of the estimate when using a 1 m std. dev. prior map with the accurate (0.1° bearing std. dev.) sensor, compared to using the line segment form as used in Figure 3.12. The map slip is still present as seen from the spikes, however

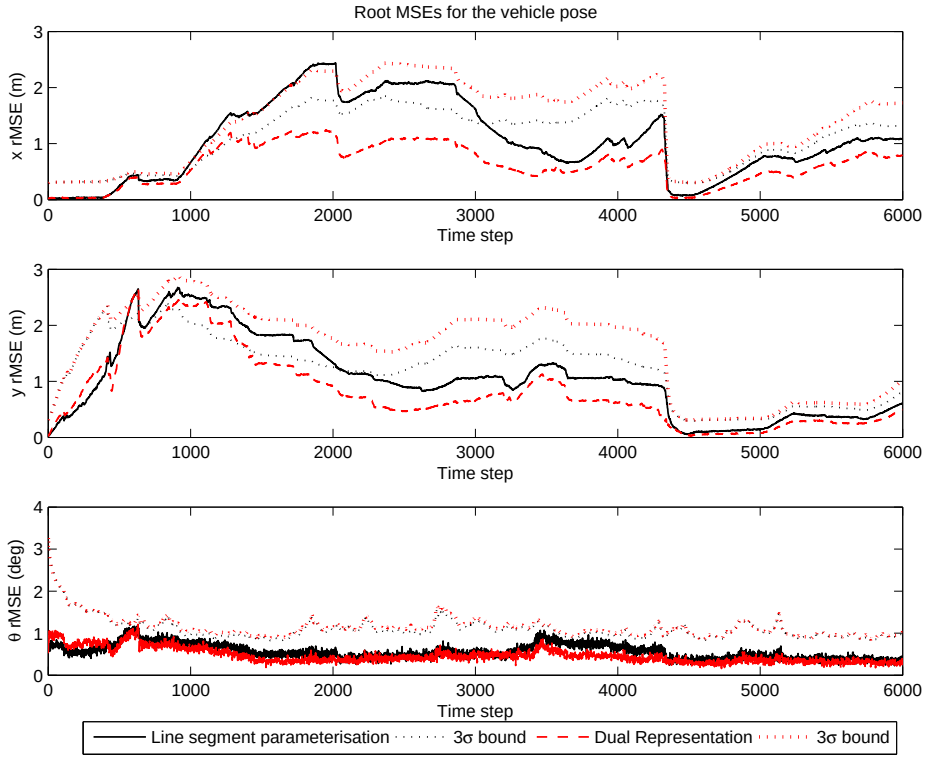

Figure 3.14: Absolute vehicle pose error for SLAM with a 2 m std. dev. prior map showing the reduced error from using the Dual Representation over the regular line segment parameterisation.

the Dual Representation appears to reduce the knock-on effect of corruption from this event, and from further errors (This is particularly noticeable for the vehicle x position).

3.6.2.3 Stabilising Noise

The structure of the Hessian indicates that the line segment form has significant second order terms, and thus we use the Second Order filter for all our experiments. If second order terms are neglected performance is degraded.

When doing simulations using the nonlinear vehicle motion and range-bearing sensor models from chapter 2 we encountered a common failure mode with the Dual Representation whereby even with the Second Order filter the map estimate would suddenly rotate off, as shown in Figure 3.16 (the line segment form without the Dual Representation does not suffer from this). We found that adding stabilising noise when using the Dual Representation mitigated this. The form of the stabilising noise we use was heuristically determined; it works within the domain of noises and models we used, however is not necessarily the required form. The stabilising noise $\delta\mathbf{R}$ is added to (3.29) to give

$$\delta\mathbf{R} = \nabla\mathbf{g}(k)\mathbf{P}(k|k-1)\nabla\mathbf{g}^T(k) \tag{3.42}$$

$$\mathbf{R}(k) = \begin{bmatrix} \mathbf{R}_p & \mathbf{R}_p^T \\ \mathbf{R}_p & \mathbf{R}_m + \delta\mathbf{R} \end{bmatrix}_k, \mathbf{R}_p = \mathbf{R}_m. \tag{3.43}$$

The vulnerability of the unstabilised Dual Representation to this failure mode appears to be a result

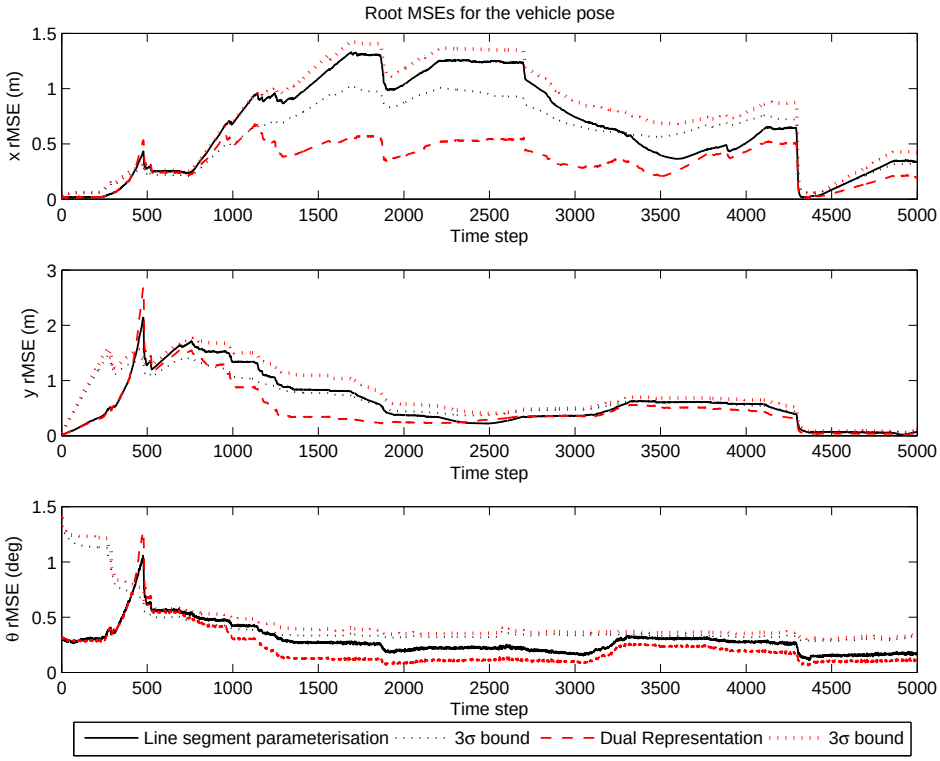

Figure 3.15: The Dual Representation reduces the degree of inconsistency compared to the line segment parameterisation when using a 1 m std. dev. prior map with an accurate (0.1° std. dev.) bearing sensor. The improvement in the x position is particularly noticeable.

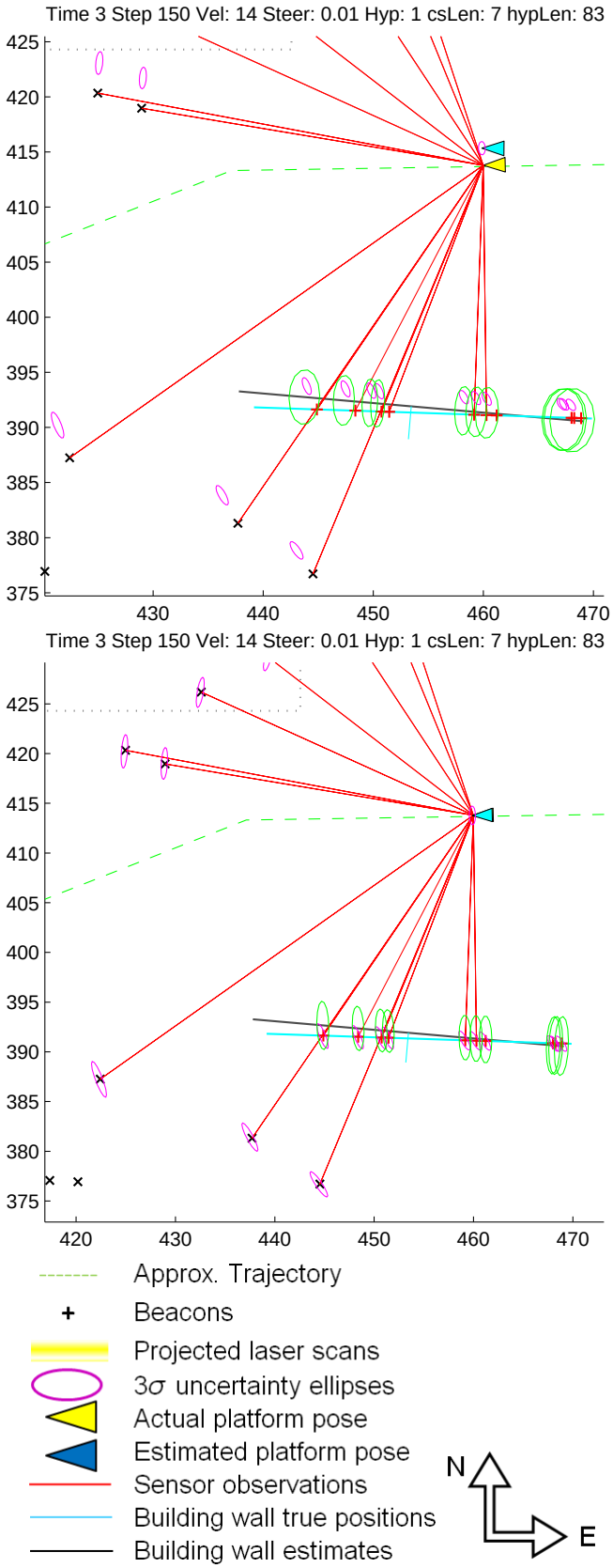

Figure 3.16: Example of map slip failure without (top) and remedied with stabilising noise (bottom) in our simulation environment (units in m).

Table 3.2: Attributes for the linear sensor used in our simulations. The range and uncertainty are the same in the x and y directions.

Parameter	Symbol	Value
Maximum range	r_{max}	50 m
Range variance	R	0.5^2 m^2
Update frequency	n/a	0.02 s

Table 3.3: Linear vehicle and environment parameters used in our simulations.

Parameter	Symbol	Value
Starting uncertainty	$\begin{bmatrix} P_x & 0 \\ 0 & P_y \end{bmatrix}_{0\|0}$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ m^2
Speed	v	14 m/s
Motion noise	R_x, R_y	$(3\triangle t)^2$ m^2
Observation time period	$\triangle t$	0.02 s
Prior map uncertainty	P_m	1 m^2

of numerics and the inability of the EKF to correctly propagate angular uncertainty, as discussed in chapter 2, rather than an error in its formulation, however further research is required to conclusively prove this. To show this we performed simulations using linear vehicle and sensor models, and the Dual Representation without stabilising noise. The trajectory covered 800 time steps (16 seconds) in our simulated urban environment. The vehicle is similar to that used in our nonlinear experiments, but has no control input or orientation errors. Instead motion noise is added to x_v and y_v. The vehicle has a linear sensor, which measures the x and y distance in global coordinates to each beacon in range, and has the parameters shown in Table 3.2. The vehicle parameters are shown in Table 3.3.

Figure 3.17 compares the vehicle pose error over the trajectory for SLAM with and without the Dual Representation, where the errors in the prior map are correlated such that the actual line segment and its estimate are known to be parallel. For comparison SLAM without a prior map is also shown. The results with the prior map are similar; the Dual Representation does not exhibit the failure mode seen in the nonlinear vehicle, indicating that without the presence of strong nonlinearity the Dual Representation is equivalent to regular SLAM with a prior map. As expected the systems that use a prior map have a more accurate pose estimate than without it.

3.6.3 Varying Prior Map Accuracy

Figure 3.18 shows that over a range of four prior map accuracies with 1σ errors ranging from 0.01 m to 1.5 m, we see the same dramatic accuracy and consistency improvement compared to SLAM without a prior map. The average error in the vehicle x and y pose estimate increases approximately linearly with increasing error in the prior map, while the average orientation error remains similar for all the prior map

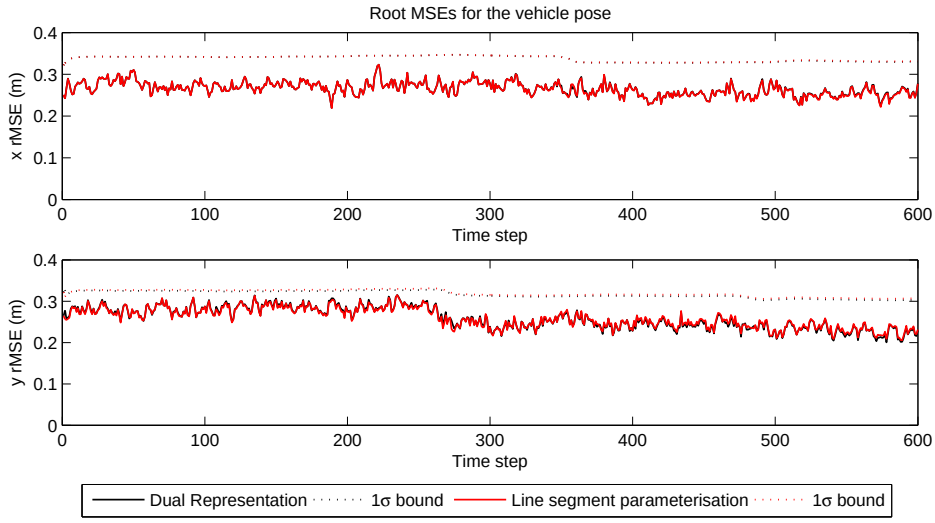

Figure 3.17: Comparison between the vehicle pose rMSEs and 1σ uncertainty bounds with and without the Dual Representation for a linear SLAM system with a nonlinear parameterised prior map. The errors in the x and y positions of each line segment are fully correlated.

accuracies (this is probably due to the steered bicycle model we use). The consistency improvement is maintained over the range of prior map accuracies.

3.7 Inferring underlying structure

The results in the previous section show that even a relatively inaccurate prior map (1.5 m error compared to the 0.01 m range error in the sensor) can dramatically improve the accuracy and consistency of SLAM in the ideal case; when the underlying structure between the features in the prior map and SLAM state are known, and thus the indicator $d(\mathbf{x}_p, \mathbf{x}_l, \{\mathbf{s}_w\})$ is known for each feature. In practice we expect that the relation status of a set of features will be unknown a priori and thus must be estimated based on the observations of the feature maps $\mathbf{z}_{pm}$, a problem we now address.

The probability $p(d_{in,s} = 1|\mathbf{z}_{pm})$, where $d_{in,s} = d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w(s)}\})$ can be computed as follows. First we define $\phi = \mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$. Thus $\theta_{in} = \phi$ if $d_{in,s} = 1$ ($p(\phi|\mathbf{x}_{pm})$ is a delta function). Then

$$\begin{aligned} p(d_{in,s} = 1|\mathbf{z}_{pm}) &= \int\int p(d_{in,s} = 1|\phi)\, p(\phi|\mathbf{x}_{pm})\, p(\mathbf{x}_{pm}|\mathbf{z}_{pm})\, \mathrm{d}\,\phi\, \mathrm{d}\,\mathbf{x}_m \\ &= \eta \int p(\phi|d_{in,s} = 1)\, p(d_{in,s} = 1)\, p(\phi|\mathbf{z}_{pm})\, \mathrm{d}\,\phi, \end{aligned} \tag{3.44}$$

where Bayes rule was used,

$$p(d_{in,s} = 1|\phi) = \eta\, p(\phi|d_{in,s} = 1)\, p(d_{in,s} = 1), \tag{3.45}$$

and

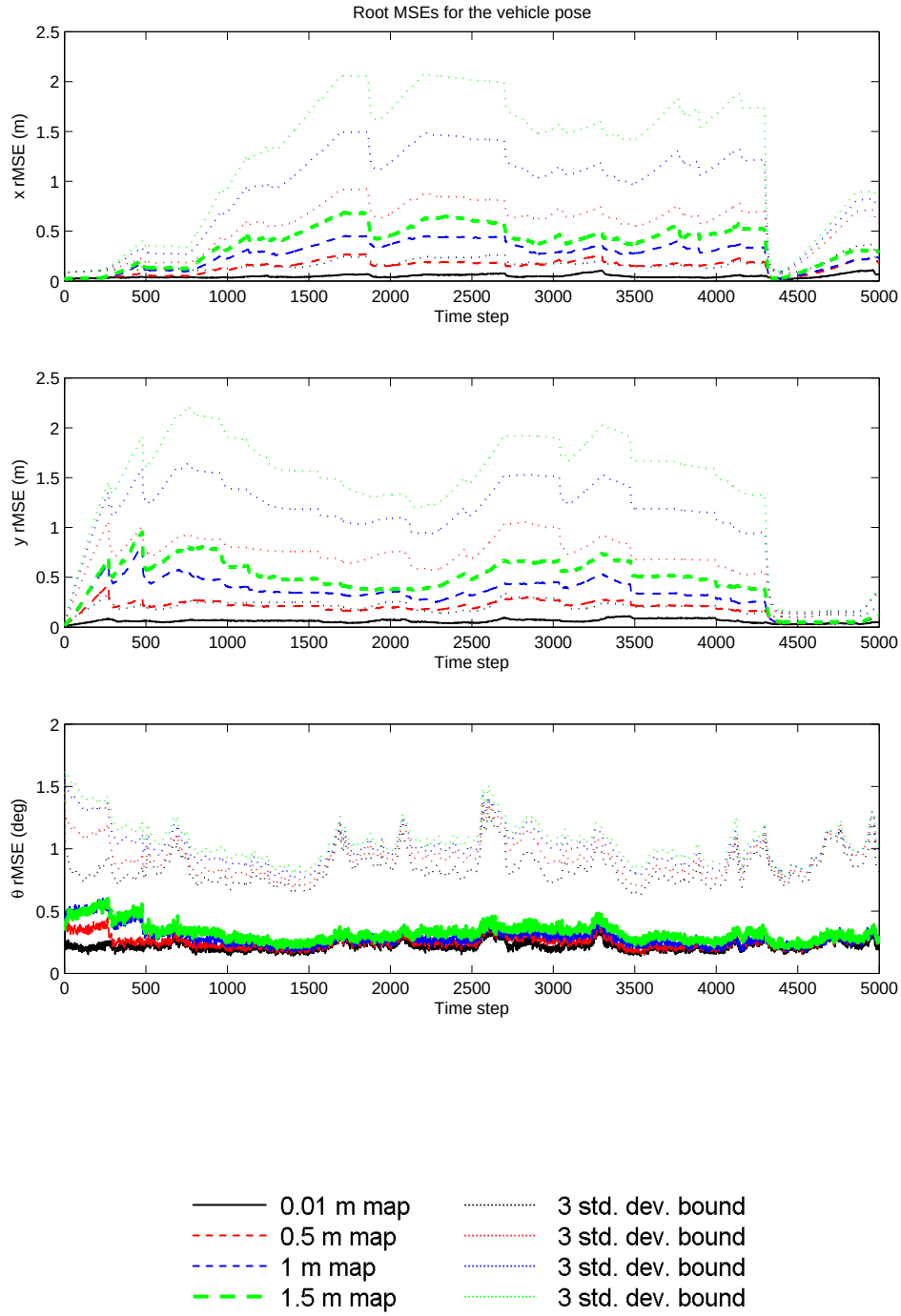

Figure 3.18: Vehicle pose estimate x, y and θ accuracy and 3σ uncertainty bounds when performing SLAM with a prior map of varying accuracy, using the Dual Representation. The legend indicates the 1σ error of each map. The consistency is similar for all the maps, with an accuracy improvement proportional to the prior map accuracy (with the exception of the orientation).

$$p(\phi|\mathbf{z}_{pm}) = \int p(\phi|\mathbf{x}_{pm})\, p(\mathbf{x}_{pm}|\mathbf{z}_{pm})\, \mathrm{d}\, \mathbf{x}_{pm}. \tag{3.46}$$

Note that $p(\phi|d_{in,s} = 1) = p(\theta_{in}|d_{in,s} = 1)$, and thus we can compute $\int p(\phi|d_{in,s} = 1)\, p(\phi|\mathbf{z}_{pm})\, \mathrm{d}\, \phi$ in this case. (3.44) is equivalent to

$$p(d_{in,s} = 1|\mathbf{z}_{pm}) = \eta\, p(\theta_{in} = \phi|d_{in,s} = 1, \mathbf{z}_{pm})\, p(d_{in,s} = 1), \tag{3.47}$$

where

$$p(\theta_{in} = \phi|d_{in,s} = 1, \mathbf{z}_{pm}) = \int p(\phi|d_{in,s} = 1)\, p(\phi|\mathbf{z}_{pm})\, \mathrm{d}\, \phi. \tag{3.48}$$

The distribution $p(\theta_{in} - \phi|d_{in,s} = 1, \mathbf{z}_{pm})$ is given by

$$p(\theta_{in} - \phi|d_{in,s} = 1, \mathbf{z}_{pm}) = p(\phi|\mathbf{z}_{pm}) \star p(\theta_{in}|d_{in,s} = 1), \tag{3.49}$$

$f \star g$ denoting cross-correlation[2] [16], and corresponds to (3.48) when evaluated at $\theta_{in} - \phi = 0$. For computing the normalising term $L(\phi = \emptyset|d_{in,s} = 0, \mathbf{z}_{pm}) = 1$ because θ_{in} is undefined in this case (i.e. we have no distribution with which to compare ϕ with).

The prior on the structure being common to the feature maps, $p(d_{in,s} = 1)$ in (3.47) can itself be conditioned on the configuration of features in one of the maps, or on other observations by exploiting a model of how the structure looks like in a particular feature map. For instance if the prior was conditioned on the configuration of features in the point map $\mathbf{x}_p$ it would have the form $p(d_{in,s} = 1|\mathbf{z})$. In chapter 5 we show how this can be done.

Having determined that $d_{in,s} = 1$ we can compute $p(\mathbf{x}_{pm}|d_{in,s} = 1, \mathbf{z}_{pm})$ by replacing $p(d_{in,s} = 1|\mathbf{x}_{pm})$ in (3.8) with the equivalent form of (3.47), where $p(d_{in,s} = 1) = 1$ giving

$$p(\mathbf{x}_{pm}|d_{in,s} = 1, \mathbf{z}_{pm}) = \eta\, p(\theta_{in} = \phi|d_{in,s} = 1, \mathbf{x}_{pm})\, p(\mathbf{x}_{pm}|\mathbf{z}_{pm}), \tag{3.50}$$

where $p(\mathbf{x}_{pm}|d_{in,s} = 1, \mathbf{z}_{pm}) = p(\mathbf{x}_{pm}|\theta_{in} = \phi, d_{in,s} = 1, \mathbf{z}_{pm})$. Thus $\mathbf{x}_p$ and $\mathbf{x}_m$ are conditioned on the consequence of $d_{in,s} = 1$, namely that $\theta_{in} = \phi$.

3.8 Summary

In this chapter we presented a probabilistic framework for exploiting prior information in SLAM. The framework is intended as a practical tool for integrating heterogeneous prior information sources with SLAM, and to probabilistically support our approach to integrating geometric prior information. Because the framework is probabilistic, it should not to be restricted to a particular technique (such as EKF-SLAM) or representation. Examples of maps that could be handled with the framework include volumetric maps, hand-drawn maps and semantic information. Potentially diverse prior information forms such as those represented by Gaussian processes (for instance kriging [93]) could be handled.

[2] $f \star g = f(-t) * g(t)$, where $*$ denotes convolution

Table 3.4: Table showing the effect of the prior map accuracy (using the Dual Representation) on the average absolute error and consistency of the vehicle pose over the entire run for the average of the Monte Carlo runs, showing that as the accuracy of the prior map increases, the platform pose estimate becomes more accurate and less inconsistent.

Prior map 1σ uncertainty	Average rMSE (m)	3σ consistency criterion (%)
0.01	0.24	89
0.5	0.36	96
1	0.55	86
1.5	0.72	82
No prior map	3.25	28

However such maps and representations are a significant research topic themselves and are outside the scope of this thesis. Thus we restrict ourselves to metric maps consisting of geometric data.

In addition, an implementation of the framework could involve multiple agents updating and sharing their maps in real time. This involves aspects of multiple-agent SLAM, data fusion and communications which are also outside the scope of this research. In this thesis we assume a "sequential" use of information; the prior map is obtained by the agent prior to performing SLAM and is not updated subsequently (other than by the SLAM process).

We also introduced the Dual Representation, a method for integrating a prior map in EKF-SLAM in such a way that the errors inherent in using the EKF are mitigated and the prior information is not corrupted. Though metric, the prior map may have a heterogeneous representation compared to the SLAM features. We have shown the benefits of the method using simulations representing a large urban environment.

When the relations between features in the prior map and the SLAM state are known, the prior map reduces the vehicle pose error and the degree of inconsistency compared to the EKF without a prior map, as summarised in Table 3.4. Even though known relations represent an ideal case, the implementation is still non-trivial, necessitating our Dual Representation approach.

In the next chapter we show how the relation indicator $d(\mathbf{x}_p, \mathbf{x}_m, \{\mathbf{s}_w\})$ in this chapter may be estimated using a greedy incremental approach to evaluating the equations in section 3.7, and present an EKF-based implementation of this for the geometric map form presented in section 3.5.

Chapter 4

Resolving Common Structure with Multiple Hypotheses

In the previous chapter we described a framework for introducing prior information into SLAM. We demonstrated the improvement that a geometric prior map can have in the ideal case; where the shared structure and thus relations between features in the state and prior map are known. However this is not true in practice. Realistically, whether a structure is shared between these features will be unknown, and must be determined. Until this is done we can only *hypothesise* as to the shared status of the structure (whether it applies or not), and attempt to decide which of these hypotheses is correct.

In this chapter we present the mathematics of incrementally determining whether a structure holds between features in the SLAM state and those in a geometric prior map, in the context of the framework developed in the previous chapter. The method works by maintaining multiple concurrent hypotheses of whether a shared structure is present or not. To this end we present a novel method, called the Common State Filter (CSF) for resolving ambiguity more efficiently than naively duplicating the entire state for each hypothesis. We evaluate the method through simulations.

4.1 Inference with Multiple Concurrent Hypotheses

Consider the case from the previous chapter where we are given a geometric prior map $p(\mathbf{x}_m|\mathbf{z}_m)$ from the observations $\mathbf{z}_m$, and then begin performing SLAM to incrementally estimate a joint map $\mathbf{x}_{p\{i\}}$ and set of platform (vehicle) poses $\mathbf{x}_v(1:k)$, $p(\mathbf{x}_v(1:k), \mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(1:k))$ conditioned on observations $\mathbf{z}_p(1:k)$. As before our example scenario consists of two maps $\mathbf{x}_p$ and $\mathbf{x}_m$ - where $\mathbf{x}_p$ is a map of salient points from a laser scanner, and the prior map $\mathbf{x}_l$ is a set of planes representing flat brick building walls ($\mathbf{s}_w$), obtained from UAV-derived aerial imagery.

Suppose the first observation of the ith feature $\mathbf{x}_{p\{i\}}$ occurs at time k_1, and we hypothesise that it has a known structure in common with the nth feature $\mathbf{x}_{m\{n\}}$. If we denote the common structure instance $\mathbf{s}_{w\{s\}}$, then the hypothesis is that $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ (we abbreviate this as $d_{in,s} = 1$ for clarity). The posterior probability (3.8) can then be expressed as

$$p(\mathbf{x}_v(k_1), \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | d_{in,s} = 1, \mathbf{z}_m, \mathbf{z}_{p\{i\}}(k_1)) = \eta\, p(\mathbf{z}_{p\{i\}}(k_1) | \mathbf{x}_v, \mathbf{x}_{p\{i\}})\, p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | \mathbf{z}_m, d_{in,s} = 1)\, p(\mathbf{x}_v(k_1)), \quad (4.1)$$

where η is a normalising term, $p(\mathbf{z}_{p\{i\}}(k_1)|\mathbf{x}_v(k_1), \mathbf{x}_{p\{i\}})$ is the observation likelihood model and

$$p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | \mathbf{z}_m, d_{in,s} = 1) = \eta\, p(\theta_{in} = \phi | d_{in,s} = 1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})\, p(\mathbf{x}_{p\{i\}})\, p(\mathbf{x}_{m\{n\}} | \mathbf{z}_m), \quad (4.2)$$

where η is a normalising term and $p(\mathbf{x}_{p\{i\}})$ is an uninformative prior. In our case $\theta_{in} = 0$ and $\phi = \mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$ from (3.11), and thus

$$p(\theta_{in} = \phi | d_{in,s} = 1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = p(\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = 0 | d_{in,s} = 1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}). \quad (4.3)$$

If the likelihood function $p(\mathbf{z}_{p\{i\}}(k_1)|\mathbf{x}_v(k_1), \mathbf{x}_{p\{i\}})$ is nonlinear (as with a range-bearing sensor), (4.1) may be difficult to reliably estimate with the EKF (similarly to the problems caused by large depth uncertainty in bearing-only SLAM [13]). This can be prevented by evaluating (4.2) instead,

$$p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | \mathbf{z}_{p\{i\}}(k_1), \mathbf{z}_m, d_{in,s} = 1) = \eta\, p(\theta_{in} = \phi | d_{in,s} = 1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})\, p(\mathbf{x}_{p\{i\}} | \mathbf{z}_{p\{i\}}(k_1))\, p(\mathbf{x}_{m\{n\}} | \mathbf{z}_m), \quad (4.4)$$

and is the method we use in our implementation.

For subsequent observations up to time k_a the estimate conditioned on $d_{in,s} = 1$ is updated as in regular SLAM,

$$p(\mathbf{x}_v(k_a), \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | d_{in,s} = 1, \mathbf{z}_m, \mathbf{z}_{p\{i\}}(k_1 : k_a)) = \eta\, p(\mathbf{z}_{p\{i\}}(k_a) | \mathbf{x}_v(k_a), \mathbf{x}_{p\{i\}})\, p(\mathbf{x}_v(k_a), \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}} | d_{in,s} = 1, \mathbf{z}_m, \mathbf{z}_{p\{i\}}(k_1 : k_a - 1)). \quad (4.5)$$

By separately maintaining an estimate of the state conditioned on $d_{in,s} = 0$, we can incrementally infer the probability (3.47) that the hypothesis $d_{in,s} = 1$ is correct as follows.

In general, for maps $\mathbf{x}$ and all observations $\mathbf{z}(k_1 : k_a)$, where d signifies the structure indicator we wish to infer, from Bayes rule

$$p(d = 1 | \mathbf{z}(k_1 : k_a)) = \left[\int \eta\, p(\mathbf{x} | d = 1)\, p(\mathbf{x} | \mathbf{z}(k_1 : k_a))\, \mathrm{d}\,\mathbf{x} \right] p(d = 1). \quad (4.6)$$

From chapter 2, the general form for updating the state based on Bayes rule is

$$p(\mathbf{x} | \mathbf{z}(k_1 : k_a)) = \eta\, p(\mathbf{z}(k_a) | \mathbf{x})\, p(\mathbf{x} | \mathbf{z}(k_1 : k_a - 1)), \quad (4.7)$$

where η is a normalising term, $p(\mathbf{z}(k_a)|\mathbf{x})$ is the observation likelihood and $p(\mathbf{x}|\mathbf{z}(k_1 : k_a - 1))$ is the posterior map estimate at time $k_a - 1$. Thus substituting this into (4.6) gives

$$p(d=1|\mathbf{z}(k_1:k_a)) = \left[\int \eta\, p(\mathbf{z}(k_a)|\mathbf{x})\, p(\mathbf{x}|\mathbf{z}(k_1:k_a-1), d=1)\, \mathrm{d}\,\mathbf{x}\right] p(d=1), \tag{4.8}$$

where η is again a normalising term such that $p(d=1|\mathbf{z})$ is a probability, and is given by

$$\eta = \sum_d \left[\int \eta\, p(\mathbf{z}(k_a)|\mathbf{x})\, p(\mathbf{x}|\mathbf{z}(k_1:k_a-1), d)\, \mathrm{d}\,\mathbf{x}\right] p(d). \tag{4.9}$$

Note how evaluating η requires us to track $p(\mathbf{x}|\mathbf{z}(k_1:k_a-1), d)$ for all possible values of d. Thus it follows that $p(d=1|\mathbf{z}(k_1:k_a))$ can be inferred in an incremental manner at time k_n by maintaining the posterior state estimates of the previous time step k_a-1 for all values of d, $p(\mathbf{x}|\mathbf{z}(k_1:k_a-1), d)$ and the observation likelihood function $p(\mathbf{z}(k_a)|\mathbf{x})$.

For the case above, from (4.1) the form of $p(\mathbf{z}(k_a)|\mathbf{x})$ in (4.8) is $p(\mathbf{z}_{p\{i\}}(k_1)|\mathbf{x}_v(k_1), \mathbf{x}_{p\{i\}})$, and the posterior is $p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})|\mathbf{z}_m, d_{in,s})\, p(\mathbf{x}_v(k_1))$.

For the first observation,

$$p(d_{in,s}=1|\mathbf{z}_{p\{i\}}(k_1), \mathbf{z}_m) = \eta\, p(\mathbf{z}_{p\{i\}}(k_1)|\mathbf{z}_m, d_{in,s}=1)\, p(d_{in,s}=1), \tag{4.10}$$

where η is a normalising term, which in this case is

$$\eta = p(\mathbf{z}_{p\{i\}}(k_1)|\mathbf{z}_m, d_{in,s}=1)\, p(d_{in,s}=1) + p(d_{in,s}=0). \tag{4.11}$$

As shown in Appendix A.1, (4.10) is equivalent to

$$p(d_{in,s}=1|\mathbf{z}_{p\{i\}}(k_1), \mathbf{z}_m) = \eta\, p(\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = 0|d_{in,s}=1, \mathbf{z}_{p\{i\}}(k_1), \mathbf{z}_m)\, p(d_{in,s}=1), \tag{4.12}$$

where $\theta_{in}=0$ and $\phi = \mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$ from our constraint form, given by (3.11).

From the above we see that we do not have to delay applying the structure constraint in order to be able to infer whether the structure is indeed shared between the two maps, however to compute the probability the structure is indeed shared from (4.8), we need to track the posterior state estimate for both hypotheses, i.e. that the structure is and is not shared.

Having applied a constraint, by evaluating (4.8) every time an observation is made, over time the probabilities of incorrect hypotheses will decrease to the point where they can be considered false, eventually revealing the correct hypothesis (that $d_{in,s}=1$ or $d_{in,s}=0$).

To implement this in an EKF-SLAM framework, we need a method of storing multiple state estimates corresponding to multiple hypotheses, evaluating (4.5) and (4.8) whenever an observation is made. To do so we use a method adopted from multiple target tracking [100], that of multiple hypothesis SLAM (MHSLAM) [106, 84].

4.2 Ambiguous Decisions and MHSLAM

MHSLAM propagates a set of m hypotheses, representing the SLAM state conditioned on all possible values for a structure $d(\mathbf{x}_{p\{I\}}, \mathbf{x}_{m\{N\}}, \{\mathbf{s}_{w\{S\}}\})$ (or set of structures) between a set of features I and N, and structure instances S. At time k each hypothesis h consists of the joint SLAM vehicle/map estimate $p(\mathbf{x}_v(k), \mathbf{x}_p | d(\mathbf{x}_{p\{I\}}, \mathbf{x}_{m\{N\}}, \{\mathbf{s}_{w\{S\}}\}) = C^h, \mathbf{z}_{m\{n\}}, \mathbf{z}_p(1:k-1))$, represented by its mean and covariance, and a weight representing the probability that the hypothesis is correct (and thus that $d(\mathbf{x}_{p\{I\}}, \mathbf{x}_{m\{N\}}, \{\mathbf{s}_{w\{S\}}\}) = C^h$). Therefore, the MHSLAM state can be written as

$$\left\{ \begin{array}{cccc} \{\hat{\mathbf{x}}, \mathbf{P}, w\}^1 & \{\hat{\mathbf{x}}, \mathbf{P}, w\}^2 & \ldots & \{\hat{\mathbf{x}}, \mathbf{P}, w\}^m \end{array} \right\}_{k|k}. \tag{4.13}$$

A normalised weighting at time k, $w^{1\ldots m}(k)$ is maintained for each filter,

$$w^h(k) \propto p(d(\mathbf{x}_{p\{I\}}, \mathbf{x}_{m\{N\}}, \{\mathbf{s}_{w\{S\}}\}) = C^h | \mathbf{z}_{p\{i\}}(1:k), \mathbf{z}_{m\{n\}}), \tag{4.14}$$

so that when normalised the weights represent the posterior probability distribution over all possible outcomes that $d(\mathbf{x}_{p\{I\}}, \mathbf{x}_{m\{N\}}, \{\mathbf{s}_{w\{S\}}\})$ could take[1].

For instance if we would like to determine whether a relation holds between the ith feature in the SLAM state and the nth feature in the prior map, two hypotheses, $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ and $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 0$ would be kept, in the hope that over time as more observations are made the posterior probability of all but one will eventually become sufficiently low that we can say that the remaining hypothesis is the correct one.

In a SLAM context this has been referred to as a "brute force" method [106], and the Gaussian Sum filter (GSF) when using multiple Gaussians to approximate another distribution [71, 3].

The steps for the algorithm are as follows:

1. Predict the state forwards in each hypothesis to the current time. This is achieved by executing the mean and covariance prediction equations (2.12) and (2.13).

2. Update and reweight. The update equations (2.18) to (2.22) are applied and the weight is updated (assuming a Gaussian likelihood)

$$w^h(k) = w^h(k-1) \left[\frac{1}{(2\pi)^{\frac{c}{2}} \sqrt{|\mathbf{S}_h|}} \exp\left(-\frac{1}{2} \nu^T \mathbf{S}^{-1} \nu \right)_h \right]_k, \tag{4.15}$$

   where, ν_h is the innovation from (2.18) with dimension c, and $\mathbf{S}_h$ is the associated innovation covariance from (2.19), in the filter representing hypothesis h. Note the recursive nature of (4.15) from (4.8).

3. Pruning. Filters with a normalised weight (posterior probability) falling below a low threshold p_{reject} (such as 0.02) are considered to correspond to false hypotheses; thus they are trimmed

[1] In practice it is not feasible to maintain the distribution over all possible outcomes indefinitely; instead the probabilities of hypotheses that become sufficiently improbable are set to 0, allowing them to be dropped from further consideration. Thus strictly speaking the distribution we maintain is an approximation.

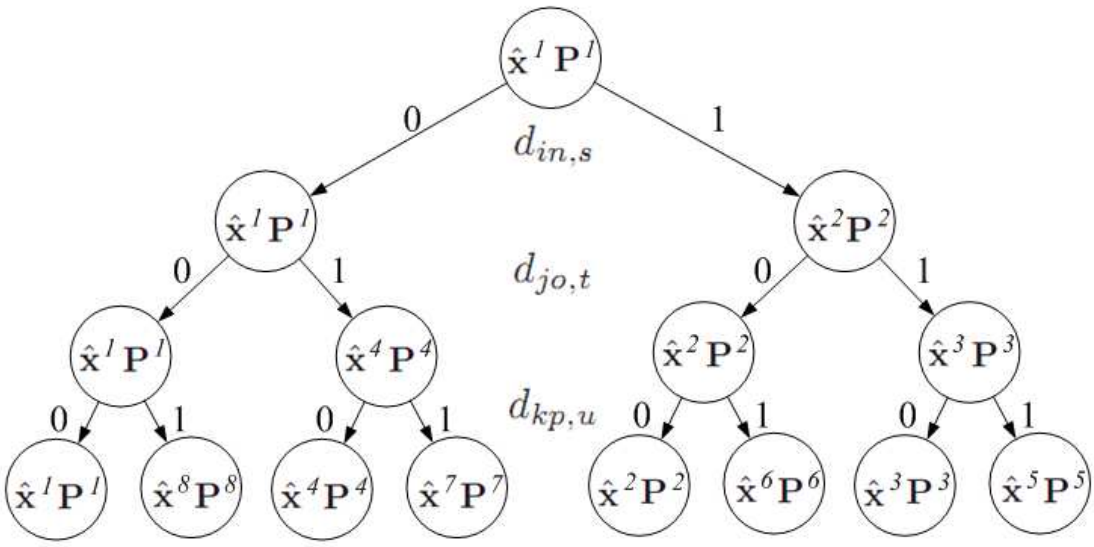

Figure 4.1: Hypothesis tree showing the explosion in the number of hypotheses as child hypotheses conditioned on the possible values of three ambiguous structure indicators $d_{in,s}$, $d_{jo,t}$ and $d_{kp,u}$ are spawned from a single parent (marginal) hypothesis.

(removed). Note that in doing so we are in effect saying that their probability is zero, and thus the posterior probabilities of remaining hypotheses will be implicitly conditioned on these hypotheses being false.

4. Spawn new hypotheses. This step is only taken when new relations are identified that need to be resolved; generally when a new beacon that could relate with a feature in the prior map is observed. New hypotheses are initialised as follows. A new beacon $\mathbf{x}_{p\{i\}}$ that may or may not obey a relation with the nth prior map feature $\mathbf{x}_{l\{n\}}$ is observed; thus the possible outcomes for this beacon are that $d_{in,s} = 1$ or $d_{in,s} = 0$. Each existing hypothesis in the MHSLAM state is thus duplicated and augmented with either one of the two possibilities, as shown in Figure 4.1, doubling the number of hypotheses in the MHSLAM state. In our case, for hypotheses conditioned on $d_{in,s} = 1$, the prior map feature is initialised into the state as in section 3.6 in chapter 3; the state for hypotheses conditioned on $d_{in,s} = 0$ remains unmodified. The weight of each hypothesis h is then given by

$$w^h(k) = w^g(k)\, L^h\, p(d_{in,s} = C^h), \tag{4.16}$$

where h was spawned from hypothesis g, and L^h is the likelihood corresponding to (4.10). This integrates the prior probability and initial likelihood of the hypothesis into the weight. Because the weights are renormalised at every time step, the distribution over weights constitutes a valid probability density function.

The initial likelihood L^h for a hypothesis h corresponds to (4.12). For the hypothesis that $d_{in,s} = 1$, in the absence of other information as to its distribution, we assume that the likelihood function is a Gaussian,

$$p(\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}) = 0 | d_{in,s} = 1, \mathbf{z}_{p\{i\}}(k), \mathbf{z}_{m\{n\}}) = \left(\frac{1}{(2\pi)^{\frac{c}{2}} \sqrt{|\mathbf{S}|}} \exp\left(-\frac{1}{2} \nu^T \mathbf{S}^{-1} \nu \right) \right)_h , \quad (4.17)$$

where c is the dimension of the innovation ν^h (which represents $\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})$), with covariance $\mathbf{S}^h$ computed from (3.21).

For subsequent observations $k+n$ the observation likelihood $p(\mathbf{z}(k+n)|\mathbf{x})$ in (4.8) is given by (4.15).

Where there are multiple hypotheses, hypothesis enumeration must be performed on each filter in turn. As the number of hypotheses spawned can grow quickly, it is important that the filter is able to detect and prune unlikely hypotheses as quickly as possible, to keep the computational and storage cost feasible. Each filter at the end of a run can be considered to correspond to a chain of hypotheses which over time have been shown to have a high likelihood of being correct. When there are multiple hypotheses $1 \ldots m$, the highest weighted state (the MAP estimate) is chosen as the representative hypothesis. In addition, if the posterior probability of this hypothesis exceeds a threshold p_{accept}, it may be accepted as the correct hypothesis and all others trimmed.

If a hypothesis requires the addition of a new beacon, it is added to the state concerned using (2.29) and (2.30), as in regular SLAM.

The main advantages of full MHSLAM over other methods is its generality, and the ability to defer association decisions while making full use of the information available [51]. In addition each hypothesis is in principle optimal. Dropping ambiguous observations as an alternative is restrictive, and leads to increased ambiguity further on [51].

However, MHSLAM has two important disadvantages. First, the number of hypotheses increase exponentially with the number of ambiguous associations. Second, a complete state must be maintained for each hypothesis. Clearly there is a large computational storage and update complexity associated with this. Even if aggressive pruning techniques are used, maintaining even a small number of hypotheses can be prohibitively expensive. If we consider a state comprising n beacons over m hypotheses, the storage requirements of full MHSLAM are $O(mn^2)$, with an update complexity of $O(mn^3)$. One way to make MHSLAM tractable is to reduce the computational and storage costs associated with each hypothesis. [109] and [70, 73] attempt to approximate MHSLAM using a single state, however because they do not maintain either full correlations or hypothesis independence, the methods either lose information while there is ambiguity or are not guaranteed to maintain consistency [17]. In the next section we present a novel algorithm that mitigates these issues, known as the Common State Filter (CSF) [96].

4.3 The Common State Filter

The intuition for the CSF is illustrated in Figure 4.2. This figure shows an example of a vehicle performing planar full MHSLAM, with the crosses representing beacons, and the covariance ellipses representing 3σ beacon estimates. There are 14 hypotheses, and the ellipses for all the beacons on the map across all the hypotheses have been superimposed (i.e. 14 ellipses are shown per beacon). Close examination shows that the map naturally segregates itself into beacons that are “near” (i.e. being observed or which

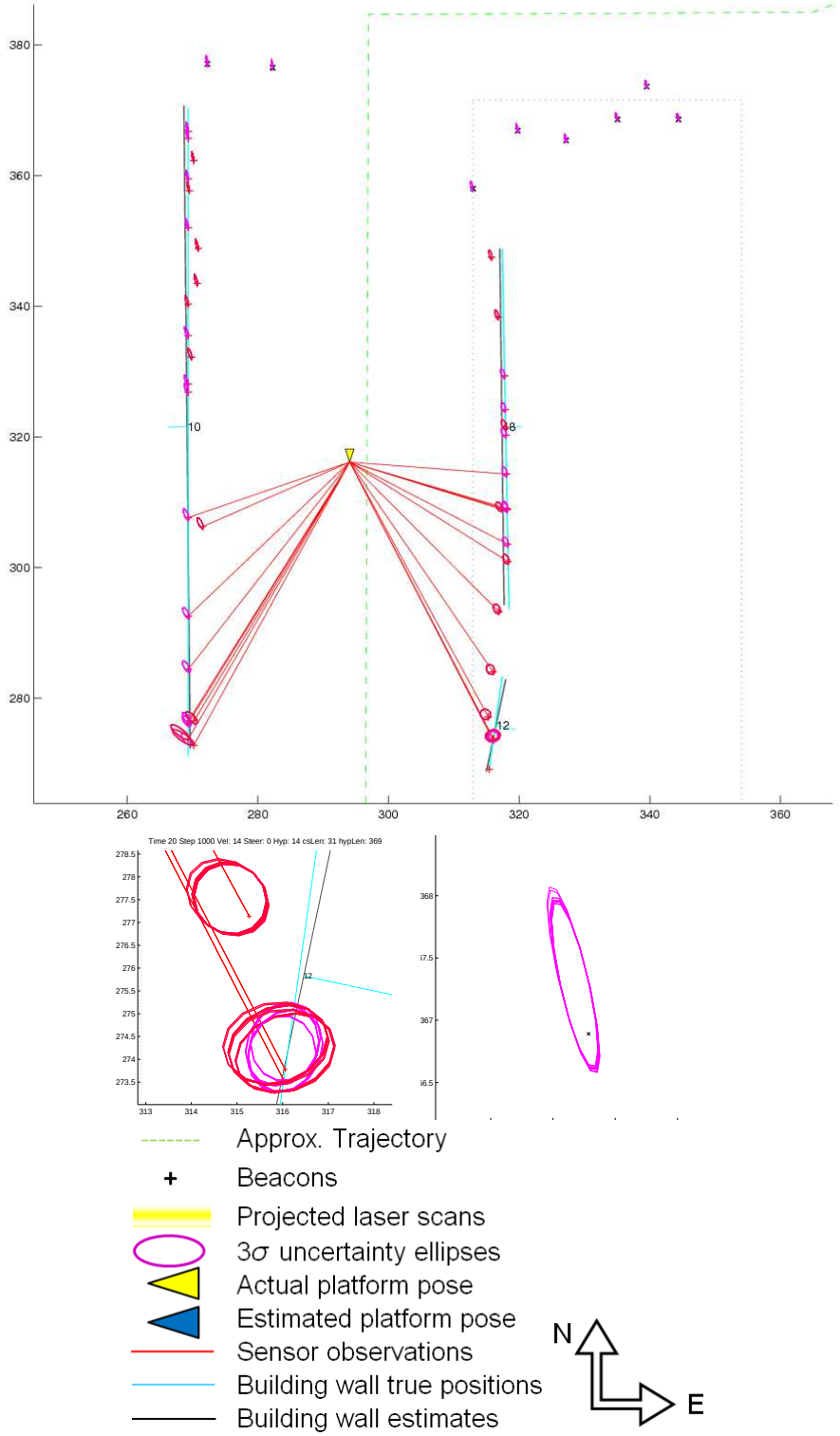

Figure 4.2: Example of a full MHSLAM run from our simluations (units in m), with 14 hypotheses. Top: The platform is moving down, and there are 14 hypotheses. Left: There is noticeable variation between groups of hypotheses close to the platform. Right: The estimates for beacons further away show less variation across the hypotheses.

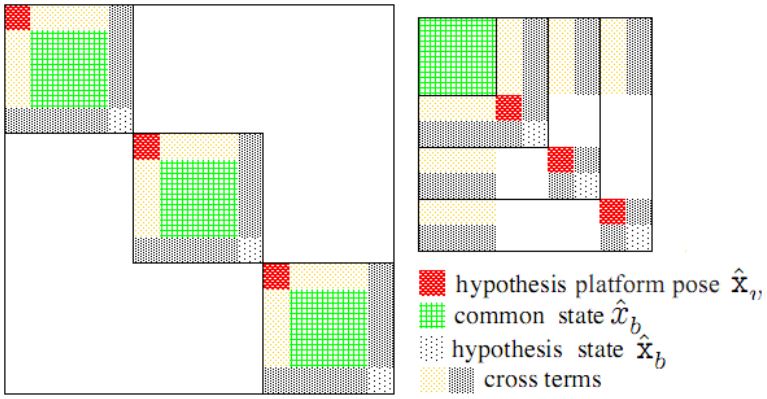

Figure 4.3: Full MHSLAM (left) and CSF (right) covariance matrices shown as a single large matrix for 3 hypotheses. The common state beacon covariances are square hatches, the hypothesis state beacons are sparse black dots. The cross terms between the platform pose and map and hypothesis covariances are shown in lighter and darker shading respectively.

have been observed recently), and "far". "Near" beacon estimates will vary substantially across hypotheses, while "far" beacons will have very similar estimates, as the map shows. Full MHSLAM makes no distinction between the two types of states and replicates the entire map across all hypotheses. This is inefficient, as it can contain hundreds of states in a large map. Based on the above intuition, the CSF removes the need to copy distant beacons across all hypotheses.

The benefit of this method is illustrated in Figure 4.3 which visually indicates the structure of the covariance matrices required to maintain MHSLAM and CSF covariances. The MHSLAM portion of the CSF is of a far smaller dimension than in full MHSLAM, and allows for greater control of storage/accuracy via the deliberate consideration of beacons within it (see section 4.4). The difference in size becomes more apparent as the number of hypotheses or the size of the state grows.

This state partitioning concept is similar to postponement [66], a method of amortising the computational costs associated with the Kalman filter, by deferring the updates of "far" beacons in the state to a more convenient time. Postponement, as applied to single hypothesis SLAM is fully optimal and is intended to give the platform increased flexibility in the times at which it updates beacons in the state. There is no storage saving associated with the method. The CSF by comparison is a suboptimal method that is mainly intended to reduce the storage and computational costs of MHSLAM at a given point in time.

In deriving the CSF let us consider the full MHSLAM structure in (4.13) above. This will become the set of hypothesis states $\left\{ \begin{bmatrix} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_b^T \end{bmatrix}^T, \mathsf{P}, w \right\}^{1\ldots m}$. We augment this with a common state $\{\hat{\mathbf{x}}_b, \mathsf{P}_b\}$, which remains the same across all hypotheses. The structure of all the hypotheses becomes

$$\hat{\mathbf{x}}^{1\ldots m}(k|k-1) = \left\{ \begin{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_b \end{bmatrix} \\ \begin{bmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{x}}_b \end{bmatrix}^1 \end{bmatrix}, \cdots, \begin{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_b \end{bmatrix} \\ \begin{bmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{x}}_b \end{bmatrix}^m \end{bmatrix} \right\}_{k|k-1} . \tag{4.18}$$

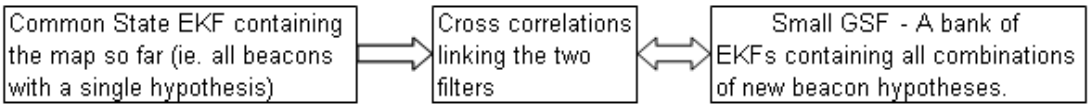

Figure 4.4: Visualisation of the Common State Filter in terms of a regular EKF and reduced order MHSLAM (the hypothesis state) linked by cross correlations. The arrows show the direction of information flow between the two filters. Information cannot flow to the common state.

The associated covariance structure of the common state and each hypothesis state, along with the cross terms between them is

$$\mathbf{P}^{1\ldots m}(k|k-1) = \left\{ \begin{bmatrix} [\,\mathtt{P}_b\,] & \left[\mathtt{P}_{bv}^T\ \mathtt{P}_{bb}^T\right]^1 \\ \left[\mathtt{P}_{vb}\ \mathtt{P}_{bb}\right]^1 & \begin{bmatrix} \mathtt{P}_v & \mathtt{P}_{vb}^T \\ \mathtt{P}_{vb} & \mathtt{P}_b \end{bmatrix}^1 \end{bmatrix}, \ldots, \begin{bmatrix} [\,\mathtt{P}_b\,] & \left[\mathtt{P}_{bv}^T\ \mathtt{P}_{bb}^T\right]^m \\ \left[\mathtt{P}_{vb}\ \mathtt{P}_{bb}\right]^m & \begin{bmatrix} \mathtt{P}_v & \mathtt{P}_{vb}^T \\ \mathtt{P}_{vb} & \mathtt{P}_b \end{bmatrix}^m \end{bmatrix}_{k|k-1} \right\}. \tag{4.19}$$

The covariance matrix of each hypothesis state is shown in uppercase, and the cross terms with the common state are shown in lowercase. Beacons that have not had a recent observation will eventually be moved into the common state, as described in section 4.3.3. The joint common and hypothesis state across all hypotheses is shown in (4.18), with corresponding covariance structure (4.19).

Because only one of the hypotheses in MHSLAM is correct, we block the information flow from the hypothesis states to the common state. To do so we use a Schmidt-Kalman update [103, 55] to ensure that information going from each hypothesis to the common state is discarded; it is here that the suboptimality arises. Figure 4.4 shows the main components of the CSF, with arrows showing the direction of permissible information flow. Note how updating the hypothesis states does not update the common state.

4.3.1 CSF prediction

The CSF prediction step applies (2.12) and (2.13) to each hypothesis in turn. In the covariance update step (2.13), the structure of the control inputs Jacobian corresponding to hypothesis h at time k, $\nabla\mathbf{G}^h(k)$ is $\left[\begin{array}{ccc} \mathbf{0} & \nabla\mathbf{G}_h^h & \mathbf{0} \end{array}\right]_k^T$, where the first term, which applies to the common state is always zero. $\nabla\mathbf{F}^h(k)$ has the form

$$\nabla\mathbf{F}^h(k) = \begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \nabla\mathbf{F}_v^h(k) & 0 \\ 0 & 0 & \mathbf{1} \end{bmatrix} \tag{4.20}$$

with respect to (4.18), where $\nabla\mathbf{F}_v^h(k)$ is the process model Jacobian for the hypothesis h. The upper left diagonal corresponding to the common state is always identity. The prediction equation (2.13) for the hypothesis h is

$$\mathbf{Q}^h(k) = \left[\begin{array}{ccc} \mathbf{0} & \nabla\mathbf{G}_h^h & \mathbf{0} \end{array}\right]_k^T \mathbf{\Sigma}(k) \left[\begin{array}{ccc} \mathbf{0} & \nabla\mathbf{G}_h^h & \mathbf{0} \end{array}\right]_k = \begin{bmatrix} \mathbf{0} & \mathtt{q}_{vb}^T & \mathbf{0} \\ \mathtt{q}_{vb} & \mathtt{Q}_{vv} & \mathtt{Q}_{vb}^T \\ \mathbf{0} & \mathtt{Q}_{vb} & \mathbf{0} \end{bmatrix}_k^h, \tag{4.21}$$

$$\mathbf{P}^h(k-1|k-1) =$$

$$= \begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \nabla\mathbf{F}_v^h(k) & 0 \\ 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} [\,\mathsf{P}_b\,] & \left[\mathsf{p}_{bv}^\intercal\ \mathsf{p}_{bb}^\intercal\right]^h \\ \left[\begin{smallmatrix}\mathsf{p}_{vb}\\ \mathsf{p}_{bb}\end{smallmatrix}\right]^h & \left[\begin{smallmatrix}\mathsf{P}_v & \mathsf{P}_{vb}^\intercal\\ \mathsf{P}_{vb} & \mathsf{P}_b\end{smallmatrix}\right]^h \end{bmatrix}_{k-1|k-1} \begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \nabla\mathbf{F}_v^h(k) & 0 \\ 0 & 0 & \mathbf{1} \end{bmatrix}^T + \begin{bmatrix} \mathbf{0} & \mathsf{q}_{vb}^T & \mathbf{0} \\ \mathsf{q}_{vb} & \mathsf{Q}_{vv} & \mathsf{Q}_{vb}^T \\ \mathbf{0} & \mathsf{Q}_{vb} & \mathbf{0} \end{bmatrix}_k^h \tag{4.22}$$

$$= \begin{bmatrix} [\,\mathsf{P}_b\,] & \left[\nabla^T\mathbf{F}_v^h(k)\,\mathsf{p}_{bv}^\intercal\ \mathsf{p}_{bb}^\intercal\right]^h \\ \left[\nabla\mathbf{F}_v^h(k)\,\mathsf{p}_{vb}\right]^h & \left[\nabla\mathbf{F}_v^h(k)\,\mathsf{P}_v\,\nabla^T\mathbf{F}_v^h(k)\ \nabla^T\mathbf{F}_v^h(k)\,\mathsf{P}_{vb}^\intercal\right]^h \\ \mathsf{p}_{bb} & \mathsf{P}_{vb}\,\nabla^T\mathbf{F}_v^h(k)\quad\ \mathsf{P}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \mathbf{0} & \mathsf{q}_{vb}^T & \mathbf{0} \\ \mathsf{q}_{vb} & \mathsf{Q}_{vv} & \mathsf{Q}_{vb}^T \\ \mathbf{0} & \mathsf{Q}_{vb} & \mathbf{0} \end{bmatrix}_k^h . \tag{4.23}$$

This equation is applied to each hypothesis state in turn. Note how the common state P_b remains unmodified, allowing it to remain common across all hypotheses.

4.3.2 CSF update

When there is a single hypothesis, the single map is updated using (2.18) to (2.22). When there are multiple hypotheses, we cannot use the standard Kalman update equations to update $\hat{\mathbf{x}}$ and $\mathbf{P}$ as we require the common state to remain the same across all hypotheses. Thus we use the Schmidt-Kalman [103][55] form of the update equations to update $\hat{\mathsf{x}}^h$ (and its corresponding P^h and p^h) for each hypothesis h without updating the common state $\hat{\mathsf{x}}$ or its covariance P.

Common state beacons which have been associated with an observation (in any hypothesis) are transferred into each hypothesis state and are updated using their respective platform poses. Moving beacons from the common state into each hypothesis state involves reordering elements; making the ith beacon with mean $\hat{\mathsf{x}}_i$ and covariance P_i explicit, the state for the hypothesis h is

$$\hat{\mathbf{x}}^h(k|k-1) = \left\{ \begin{bmatrix} \begin{bmatrix} \hat{\mathsf{x}}_b \\ \hat{\mathsf{x}}_i \end{bmatrix} \\ \begin{bmatrix} \hat{\mathsf{x}}_v \\ \hat{\mathsf{x}}_b \end{bmatrix}^1 \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} \mathsf{P}_b & \mathsf{P}_{bi}^T \\ \mathsf{P}_{ib} & \mathsf{P}_i \end{bmatrix} & \left[\begin{smallmatrix}\mathsf{p}_{bv}^\intercal & \mathsf{p}_{bb}^\intercal\\ \mathsf{p}_{iv}^\intercal & \mathsf{p}_{ib}^\intercal\end{smallmatrix}\right]^h \\ \left[\begin{smallmatrix}\mathsf{p}_{vb} & \mathsf{p}_{vi}\\ \mathsf{p}_{bb} & \mathsf{p}_{bi}\end{smallmatrix}\right]^h & \left[\begin{smallmatrix}\mathsf{P}_v & \mathsf{P}_{vb}^\intercal\\ \mathsf{P}_{vb} & \mathsf{P}_b\end{smallmatrix}\right]^h \end{bmatrix} \right\}_{k|k-1}^h \tag{4.24}$$

before the beacon is moved, and

$$\mathbf{P}^h(k|k-1) = \left\{ \begin{bmatrix} \begin{bmatrix} \hat{\mathsf{x}}_b \\ \hat{\mathsf{x}}_v \\ \hat{\mathsf{x}}_b \\ \hat{\mathsf{x}}_i \end{bmatrix}^1 \end{bmatrix}, \begin{bmatrix} [\,\mathsf{P}_b\,] & \left[\mathsf{p}_{bv}^\intercal\ \mathsf{p}_{bb}^\intercal\ \mathsf{P}_{bi}^T\right]^h \\ \left[\begin{smallmatrix}\mathsf{p}_{vb}\\ \mathsf{p}_{bb}\\ \mathsf{P}_{ib}\end{smallmatrix}\right]^h & \left[\begin{smallmatrix}\mathsf{P}_v & \mathsf{P}_{vb}^\intercal & \mathsf{p}_{vi}\\ \mathsf{P}_{vb} & \mathsf{P}_b & \mathsf{p}_{bi}\\ \mathsf{p}_{iv}^\intercal & \mathsf{p}_{ib}^\intercal & \mathsf{P}_i\end{smallmatrix}\right]^h \end{bmatrix} \right\}_{k|k-1}^h \tag{4.25}$$

after. This operation is performed for every hypothesis. The update operation can now take place.

As all beacons with observations will be found in the hypothesis state, any non-zero terms in the observation Jacobian $\nabla\mathbf{H}$ (corresponding to the observation function (2.6)) will be confined to entries in the hypothesis states, so we need only consider this part of the Jacobian, $\nabla\mathbf{H}^h$. The gain $\mathbf{K}^h(k)$, which corresponds to the h hypothesis state is computed from the standard Kalman or Second Order filter update equations in section 2.8 of chapter 2. The hypothesis state and covariance are updated by

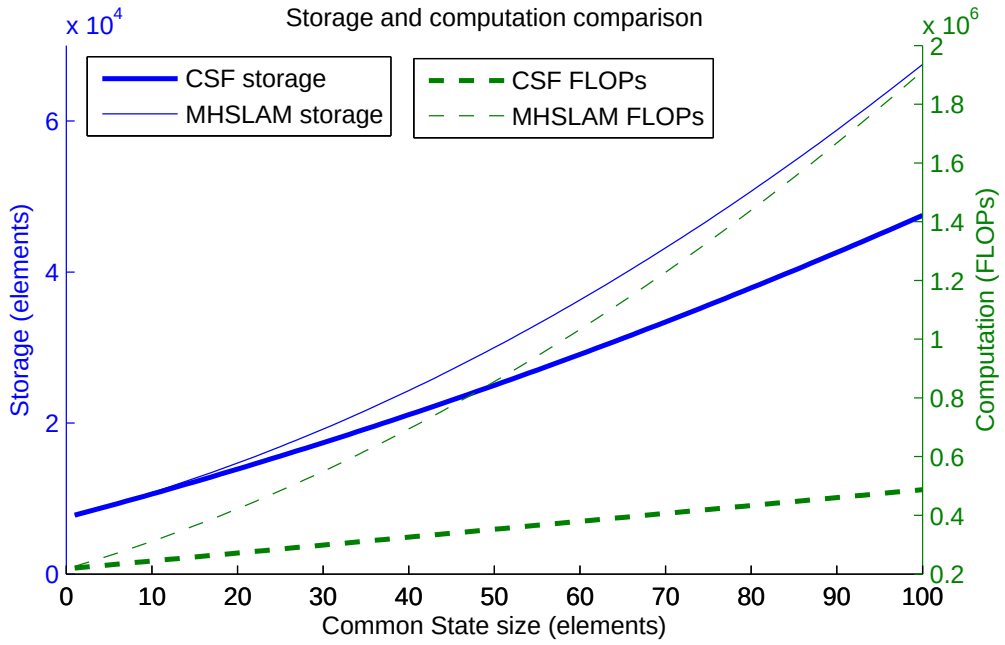

Figure 4.5: Covariance storage and computational requirements for the CSF and full MHSLAM as the state size increases. This assumes the hypothesis states have 50 elements, and that there are 5 hypotheses. For 2D point beacons, 50 elements corresponds to 25 beacons, or 16 in Dual Representation form (for a line segment prior map).

$$\begin{bmatrix} \hat{\mathrm{x}} \\ \hat{\mathrm{x}}^h \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{x}} \\ \hat{\mathrm{x}}^h + \mathbf{K}^h(k)\,\widetilde{\nu}^h(k) \end{bmatrix} \tag{4.26}$$

$$\mathbf{J} = \mathbf{I} - \mathbf{K}^h(k)\,\nabla\mathbf{H}^h(k), \tag{4.27}$$

$$\mathbf{N} = \mathbf{K}^h(k)\,\mathbf{R}(k)\,\left(\mathbf{K}^h(k)\right)^T \tag{4.28}$$

$$\begin{bmatrix} \mathrm{P} & (\mathrm{p}^h)^T \\ \mathrm{p}^h & \mathrm{P}^h \end{bmatrix}_{k|k} = \begin{bmatrix} \mathrm{P} & \mathbf{J}^T\,(\mathrm{p}^h)^T \\ \mathbf{J}\,\mathrm{p}^h & \mathbf{J}\,\mathrm{P}^h\,\mathbf{J}^T \end{bmatrix}_{k|k-1} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{N} \end{bmatrix}_k^h, \tag{4.29}$$

where $\widetilde{\nu}^h(k)$ is the innovation $\nu^h(k)$ in the regular Kalman update, or $\widetilde{\nu}^h(k) = \nu^h(k) - \frac{1}{2}(\mathrm{P}^h(k|k-1)\,\mathbf{M}^h(k))$ in the Second Order filter update, $\mathbf{M}^h(k)$ being the Hessian of the observation equation with Jacobian $\nabla\mathbf{H}^h(k)$.

Because the common state P is not updated, there is a computational improvement over the standard Kalman update used by full MHSLAM.

Following the update stage, the weight of each hypothesis is updated and trimmed based on (4.15), in the same way as in full MHSLAM.

New beacons are appended to their respective hypothesis states in the same way as in regular MHSLAM, using (2.29) and (2.30). It follows that if there is a single hypothesis, beacons will be appended to the single common state.

The CSF reduces the full MHSLAM storage and update complexity to $O(n(m+n))$ for storage

and $O(n(m+n^2))$ for the update. Figure 4.5 compares the storage requirements of full MHSLAM and the CSF for a typical scenario, showing how even for a small number of hypotheses the storage savings of the CSF over full MHSLAM become readily apparent. This efficiency comes at a penalty; there is an information loss associated with this simplification, however as we will show in Section 4.5.3 we have found this loss to be small in practice.

The update presented here requires common state beacons to be moved into each hypothesis state before they can be updated. This results in a storage and computational increase; in the Appendix we describe a less optimal alternative method that may suffice in certain applications.

4.3.3 Merging

Once all except one hypothesis have been trimmed[2], the hypothesis state beacons are moved into the common state. If we consider the remaining hypothesis state c, then just prior to merging, the state and covariance appear as follows;

$$\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathtt{x}}_b^T & \begin{bmatrix} \hat{\mathtt{x}}_v^T & \hat{\mathtt{x}}_b^T \end{bmatrix}^c \end{bmatrix}^T_{k|k}, \tag{4.30}$$

$$\mathbf{P}(k|k) = \begin{bmatrix} [\mathtt{P}_b] & \begin{bmatrix} \mathtt{p}_{bv}^T & \mathtt{p}_{bb}^T \end{bmatrix}^c \\ \begin{bmatrix} \mathtt{p}_{vb} \\ \mathtt{p}_{bb} \end{bmatrix}^c & \begin{bmatrix} \mathtt{P}_v & \mathtt{P}_{vb}^T \\ \mathtt{P}_{vb} & \mathtt{P}_b \end{bmatrix}^c \end{bmatrix}_{k|k}. \tag{4.31}$$

Moving the platform pose to the top, the merged state is

$$\hat{\mathbf{x}}'(k|k) = \begin{bmatrix} \hat{\mathtt{x}}_v^T & \hat{\mathbf{x}}_b^T & \hat{\mathtt{x}}_b^T \end{bmatrix}^T_{k|k}, \tag{4.32}$$

and the covariance

$$\mathbf{P}'(k|k) = \begin{bmatrix} \mathtt{P}_v & \mathbf{p}_T^{vb} & \mathtt{P}_{vb}^T \\ \mathbf{p}_{vb} & \mathbf{P}_b & \mathtt{p}_{bb}^T \\ \mathtt{P}_{vb} & \mathtt{p}_{bb} & \mathtt{P}_b \end{bmatrix}_{k|k}. \tag{4.33}$$

4.4 Common State Information Management

On one hand, it is beneficial to place beacons in the hypothesis states as the full hypothesis state and covariance is updated for any beacon observations. On the other hand, placing a beacon in the hypothesis state removes the storage and computational benefits that the common state provides. Taking this to the limit, if all the beacons are placed in the hypothesis state, the CSF becomes full MHSLAM. Thus ideally we would like to compromise, and identify a subset of beacons in the common state that would provide the most accuracy benefit if updated, and move them into the hypothesis states.

We can determine which beacons are significant enough to place in the hypothesis state by considering the mutual information (MI) [26] between each beacon in the common state and each hypothesis

[2] Beacons could also be merged and thus moved into the common state even when there are multiple hypotheses, however doing so in a mathematically rigorous way remains the subject of further work.

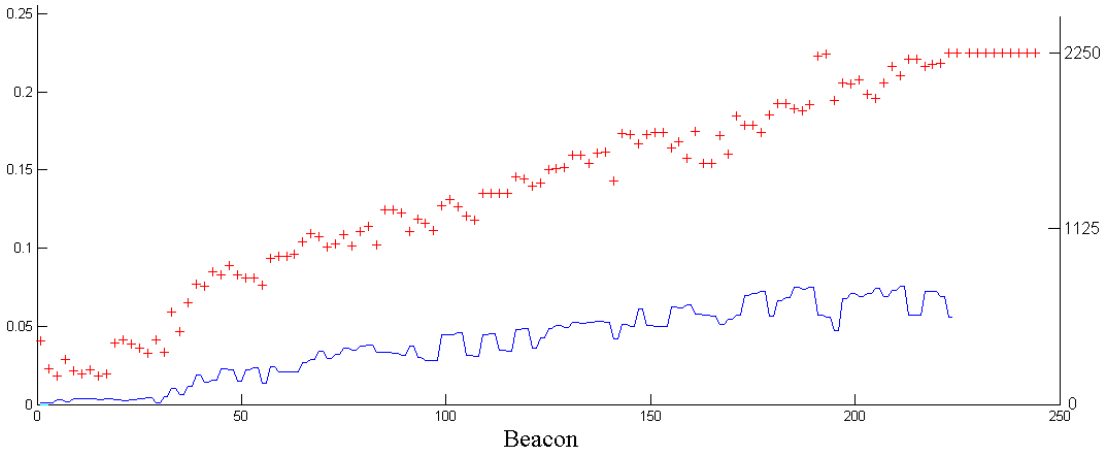

Figure 4.6: (Top) Time step at which a particular beacon in the state was last observed (the current time step is 2250). (Bottom) Euclidean distance of the posterior beacon estimates from their prior estimates following a Kalman update. In general, the greater the time elapsed since a beacon was seen, the less its estimate is affected by a Kalman update of recently seen beacons.

state platform. The MI captures the strength of the correlations between the platform poses and beacons in the common state without any spatial or temporal assumptions and is thus general. The MI between the h hypothesis platform pose $\mathrm{P}_{\mathrm{v}}^{\mathrm{h}}$ and each beacon b in the common state is given by

$$\mathbf{I}(\mathsf{P}_b, \mathrm{P}_{\mathrm{v}}^{\mathrm{h}})_{k|k-1}^{h} = \frac{1}{2}\log_2\left(\frac{|\mathsf{P}_b(k|k-1)|}{|\mathsf{P}_b(k|k-1) - \mathrm{p}_{\mathrm{bv}}^{\mathrm{h}}(k|k-1)\,\mathrm{P}_{\mathrm{v}}^{\mathrm{h}}(k|k-1)^{-1}\,\mathrm{p}_{\mathrm{bv}}^{\mathrm{h}}(k|k-1)^{T}|}\right). \quad (4.34)$$

This tells us the extent to which updating the hypothesis platform pose would update the beacon. If a beacon in the common state has a high MI with a hypothesis state pose, its estimate can potentially change significantly, so it should be moved. We move a beacon from the common state if $\mathbf{I}(\mathsf{P}_b, \mathrm{P}_{\mathrm{v}}^{\mathrm{h}})_{k|k-1}^{h} > I_{thresh}$ for any hypothesis h, accuracy taking preference over performance with decreasing I_{thresh}.

The generality of this measure means we can expect it to give good performance in a variety of situations, including loop closure. However it can be expensive to compute, and thus in trajectories without loop closure a simpler time-based metric, described below can give similar performance.

Consider a platform performing SLAM. Over time, as the platform noise accumulates, its correlation with beacons left behind decreases, and thus they become less significant. Figure 4.6 shows a plot of the amount that the mean of each beacon in the state has translated following a Kalman update of a small subset of those beacons (the latest ones seen). There is an obvious trend whereby the longer the time elapsed since the beacon was last seen, the less its position estimate changes[3]. Therefore, a crude but straightforward metric to determine which beacons are significant is to transfer those beacons which were observed within the last k_{thresh} time steps, a greater k_{thresh} favouring accuracy over performance. We used this metric in our simulations as we found it to give similar performance to the more rigorous MI approach, while being computationally cheaper. We use a value of $k_{thresh} = 1.5\,\mathrm{s}$.

[3]Here time is an approximation; the relationship is spatial, however this depends on the robot control policy [84].

Before the update stage, provided there are multiple hypotheses, all beacons in the common state are tested with the chosen metric, and those above a certain threshold (whose value is chosen with regard to the storage and accuracy trade off) are placed into each hypothesis state, where they reside along with the beacons currently being observed, until a single hypothesis remains, where they are integrated into the common state according to Section 4.3.3.

As mentioned, the worst-case storage and computational cost in this case occurs when all the beacons are moved into the hypothesis states, in which case the CSF behaves as full MHSLAM, with the same computational and storage costs.

4.5 Experiments

Because of the computational complexity of EKF-SLAM and the lack of map advanced map management in our Matlab implementation, we only performed the MHSLAM experiments over 1000 time steps of the 5000 time step trajectory (the total covariance update cost in terms of FLOPs alone being 5 times greater than regular single-state SLAM over the same trajectory). However as we are comparing the computational complexity of the CSF with full MHSLAM, unlike in chapters 2 and 3 we do not remove any beacons from the state.

As the cost associated with initialising hypotheses, computing Jacobians and updating the likelihoods for each hypothesis took a significant additional amount of time, thus we only performed 50 Monte-Carlo runs, using identical noises for all the algorithms for a given run.

4.5.1 Modelling Clutter

While some beacons in our environment lie on the walls, some lie close to walls but not on them. In their case the structure does not hold, and thus should not be used. We term such beacons *clutter*, because although they are useful features for SLAM, they cannot exploit information from the prior map. Attempting to exploit a relation that does not hold (i.e. the structure is not shared) with such beacons will lead to an inconsistent estimate.

We simulate clutter as follows. At the start of each run, a given percentage (or *clutter density*) of the beacons on all the walls are randomly displaced normal to the wall to create clutter. Beacons that are clutter always lie in front of their walls (i.e. outside of the building) at a uniformly distributed distance between 1 mm and 5 standard deviations of the variance P_m in the prior map covariance matrix, as shown in Figure 4.8. For the MHSLAM experiments in this chapter $P_m = 0.5^2\ \mathrm{m}^2$. This models the effect of features in front of building walls, such as railings, trees and signs. In the absence of real clutter distribution data, we hypothesise that this distribution is sufficiently realistic to show the algorithm working.

Figure 4.7 shows a typical line segment showing non-clutter and clutter beacons.

4.5.2 The Clutter Prior

The prior in (4.12), $p(d_{in,s} = 1)$ is computed from (3.5). We use a constant prior for $p(d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1)$, the probability that wall features come from walls (encompassing spurious features), where the kth wall structure instance is defined in terms of producing line $\mathbf{x}_{m\{n\}}$. We

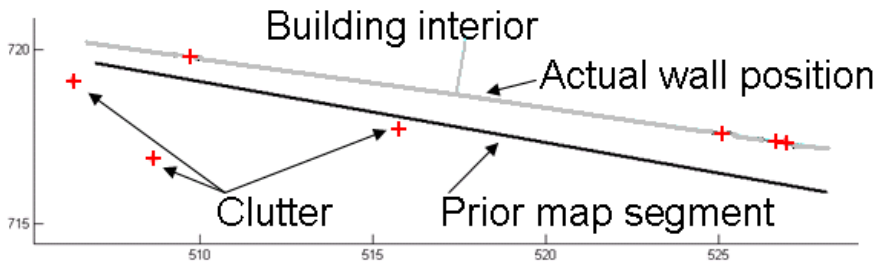

Figure 4.7: Close up of a typical line segment in the map. The crosses show beacons; some lie on the wall and are thus non-clutter. The grey line is the actual wall position, with the building interior indicated by the short normal line. The black line shows the noisy prior map segment from the prior map.

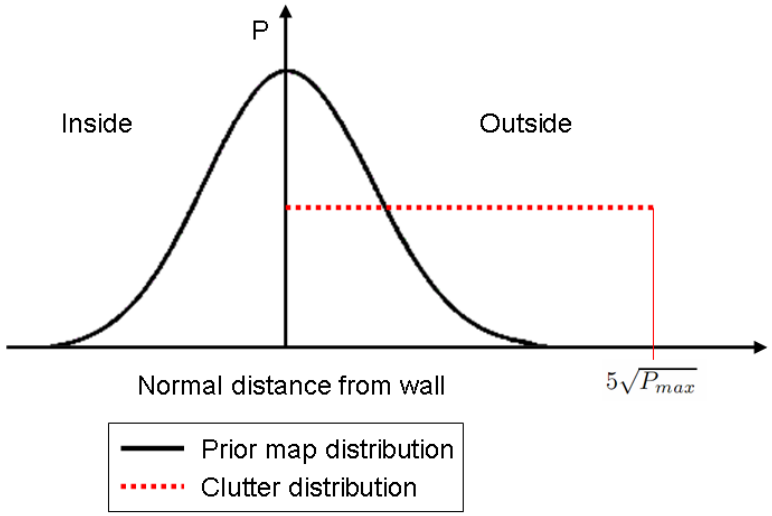

Figure 4.8: Prior map and clutter distribution normal to a wall. The clutter is distributed between 1 mm and $5\sqrt{P_{max}}$ m in front of the wall (corresponding to outside the building), where P_{max} is the variance of each element in the prior map.

also use a constant for $p(d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\}) = 1 | d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1)$, the probability that the point $\mathbf{x}_{p\{i\}}$ came from the structure instance s defined as having generated the line segment $\mathbf{x}_{m\{n\}}$ (with $p(d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\}) = 1 | d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 0) = 0$). Furthermore we assume that the structure instance heritage of independent features is independent.

Generalising to the set of features $\mathbf{x}_{p\{I\}}$ and $\mathbf{x}_{m\{N\}}$, which share the common structure instances $\mathbf{s}_{w\{S\}}$, $P(d_{IN,S} = C)$ can be inferred from the structure of the hierarchy as products of these density functions, considering that

$$P(d_{IN,S} = C) = p(d_{I,S} = A | d_{N,S} = B)\, p(d_{N,S} = B), \tag{4.35}$$

where $A\, B = C$, and A, B, C are binary vectors of 1s and 0s.

For instance for the set of J points $\mathbf{x}_{p\{J\}}$ that may come from the same wall structure instance $\mathbf{s}_{w\{s\}}$ as the the nth line segment $\mathbf{x}_{m\{n\}}$,

$$p(d_{Jn,s} = D) = \begin{cases} p(d_{n,s} = 0) + p(d_{n,s} = 1) \prod_{j \in J} p(d_{j,s} = 0 | d_{n,s} = 1) & \text{if } D = \mathbf{0} \\ p(d_{n,s} = 1) \prod_{j \in J} p(d_{j,s} = D^j | d_{n,s} = 1) & \text{otherwise} \end{cases}, \tag{4.36}$$

where D is a binary vector of 1s and 0s.

4.5.3 Results

Figure 4.9 compares the platform pose error over the trajectory for the 30 averaged Monte Carlo runs. The CSF estimate is indistinguishable from the full MHSLAM estimate, at a total covariance update cost over the run of 40% that of full MHSLAM. In all cases the average error is well within the 3σ uncertainty bound, as expected given our findings on the effect of a prior map in the previous chapter.

Figure 4.10 confirms that there is no appreciable difference in consistency levels between the CSF and full MHSLAM. At 3σ 80% of the runs met our consistency criterion, compared to less than 30% for regular SLAM without a prior map.

Figure 4.11 shows that the number of correct and incorrect constraints applied are similar for the CSF and full MHSLAM. However, the number of incorrect constraints applied is greater when the Dual Representation is not used. This suggests that the EKF and linearisation errors that the Dual Representation reduces degrade our ability to distinguish the correct hypothesis. The number of correct relations applied on average by the end of the trajectory is almost 40, whereas on average only 1.6 incorrect relations are made. EKF and errors from the nonlinearity of the models and prior line segment form all contribute to incorrect relation hypotheses being applied. However, provided this number is small compared to the number of correct relations made, the effect of these is small.

The incremental MHSLAM approach appears to be effective at distinguishing whether relations hold, and is able to apply around 80% of the maximum number of relations that could be applied. The Dual Representation works with this framework and yields a modest reduction in the false positive rate, with a corresponding decrease in the platform pose error.

Figure 4.12 shows that the number of hypotheses present over the trajectory for the CSF and full

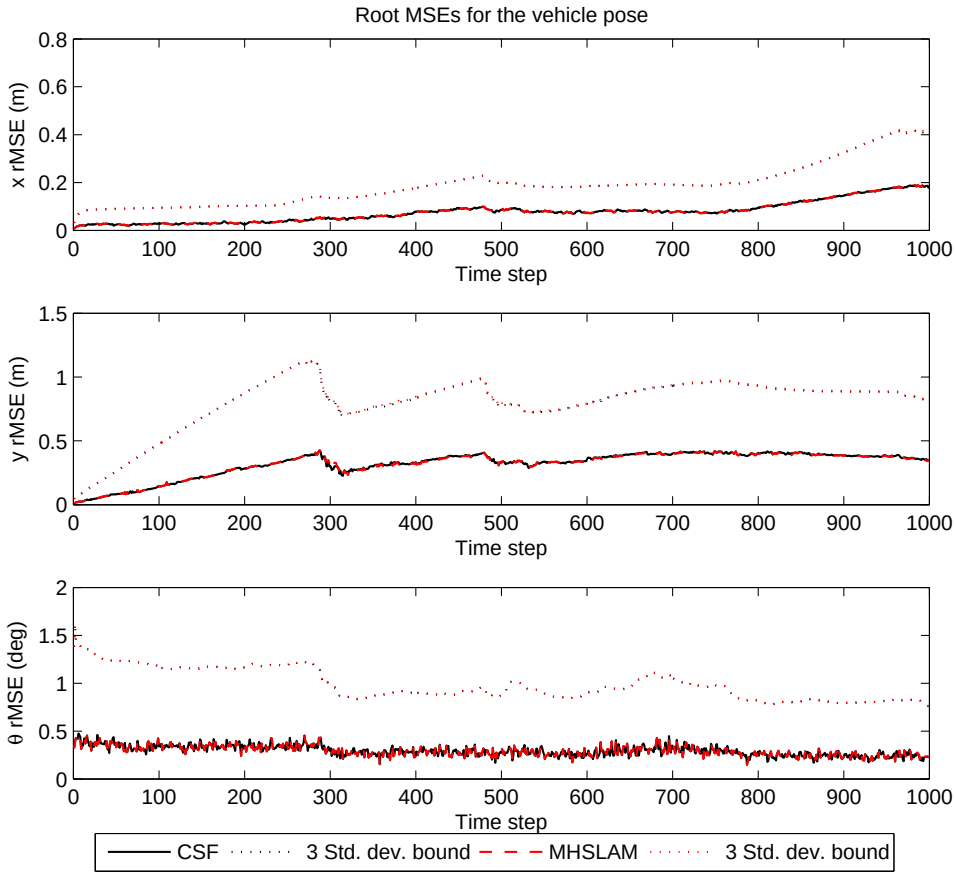

Figure 4.9: Platform pose error over the trajectory for the CSF and full MHSLAM, showing how the CSF gives effectively the same results as MHSLAM.

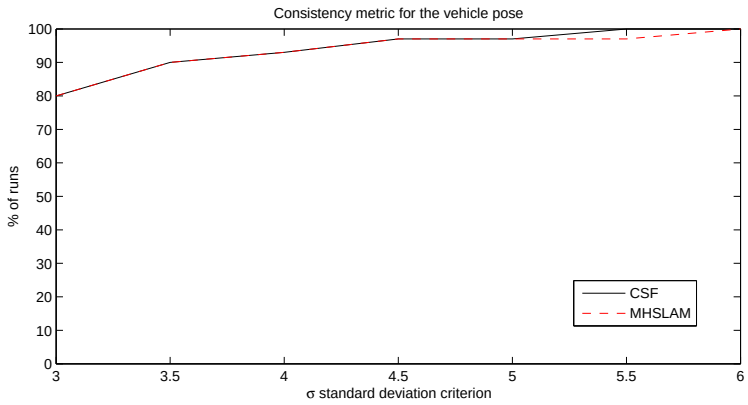

Figure 4.10: Consistency metric indicating for what percentage of runs the vehicle pose was within a given standard deviation criterion for at least 95% of the run, for the CSF and full MHSLAM. Both runs have almost identical consistency levels; the difference at the 5.5σ criterion consistutes 1 run (out of the 30 total) so does not constitute a significant difference.

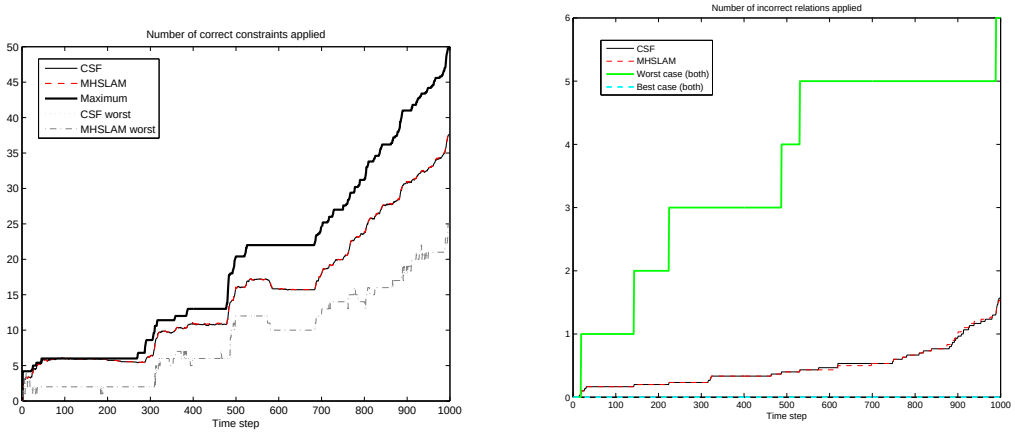

Figure 4.11: Left: The average number of correct relations applied over the 30 MC runs compared to the maximum number that could be applied at this clutter level (30%), and the worst case run (least number applied). Right: The average number of relations applied which do not hold (false positives), along with the worst and best case runs. Both the CSF and MHSLAM best and worst cases were the same. By the end of the run less than 2 incorrect relations (on average) were made. In the best case no incorrect relations were applied.

MHSLAM are almost identical, suggesting that the posterior distribution and its evolution over time in the CSF represents that of full MHSLAM.

Figure 4.13 shows the evolution of the posterior probabilities of the non-trivial hypotheses generated by one of the runs, along with the number of hypotheses. Over the entire run there were 389 significant hypotheses generated (these were not immediately trimmed after initialisation); they were spawned and trimmed at various times over the run. They are represented by the y axis, and their probability evolution at each time step (the x axis) by a grayscale value representing the probability. This shows how hypotheses are trimmed at various time steps (their trail abruptly ends), and how most trimming occurs soon after the observation of new beacons with ambiguous relations (and thus the creation of new hypotheses), as observed at various time steps including 300, 490 and 680. This occurs because the sensor is sufficiently accurate that the posterior probability of a hypothesis converges to its final value quickly, so improbable hypotheses are quickly trimmed. However if the combined platform pose and prior map uncertainty is high enough there is not enough information to give a clear winner. The introduction of a new relation introduces new information through the correlations of its features, thus dramatically altering the posterior probability distribution.

Figure 4.14 shows the posterior probability of the maximum a priori (MAP) estimate over time. Note how the greater the number of hypotheses the lower the probability of the MAP hypothesis. In general the probability of incorrect hypotheses will decrease to the point they can be rejected, however the probability a single hypothesis will generally not be high enough to accept it and reject all others.

Figures 4.16 and 4.15 show the approximate computational cost of storing and updating all the hypotheses. Only the Kalman update cost is considered (the cost of computing the Jacobians, second

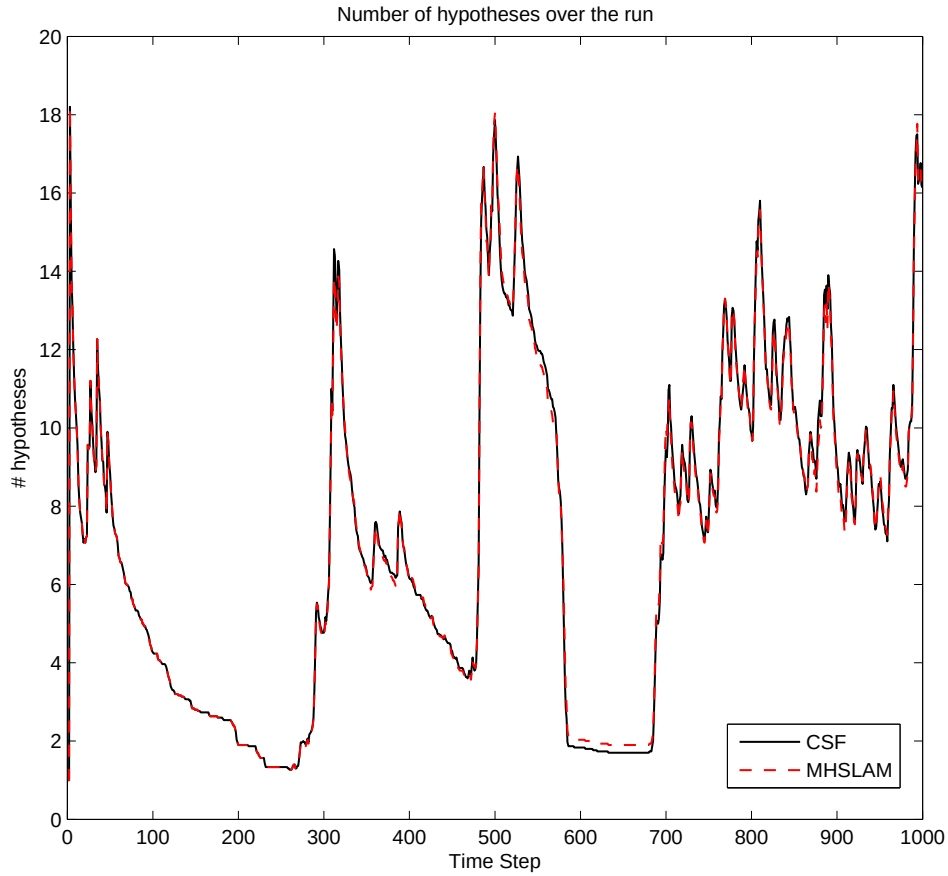

Figure 4.12: Average number of hypotheses over the trajectory for the CSF and full MHSLAM.

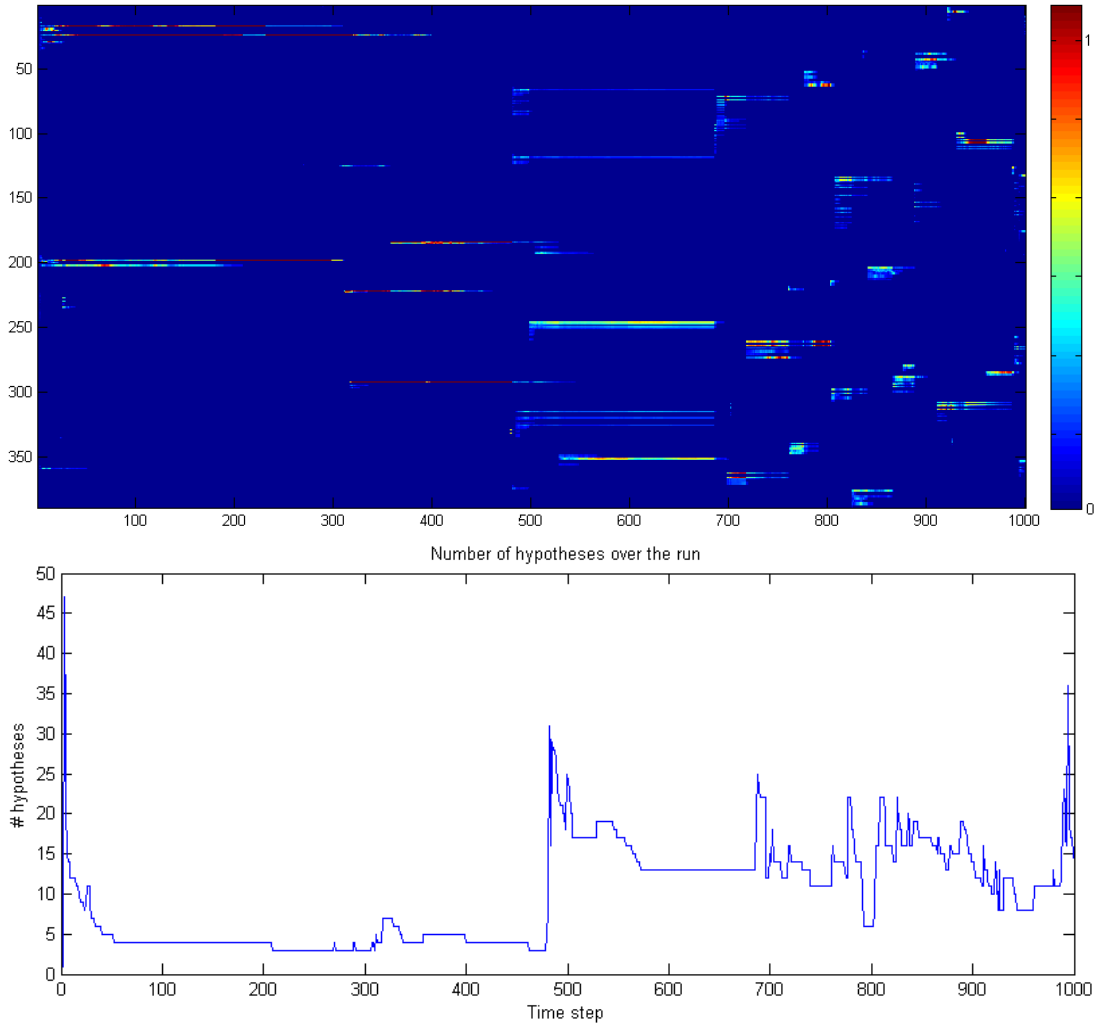

Figure 4.13: Top: Evolution of the posterior probabilities of the 389 significant hypotheses generated throughout a run. The probability of each hypothesis at each time step is represented by a colourscale value between red (p=1) and blue (p=0). The probabilities for hypotheses that have not been initialised yet are also shown as blue. Thus a cross-section of the diagram at any time step represents the posterior distribution over the hypotheses. Bottom: The number of hypotheses present in the MHSLAM state at every time step (those hypotheses whose probability is greater than 0.02).

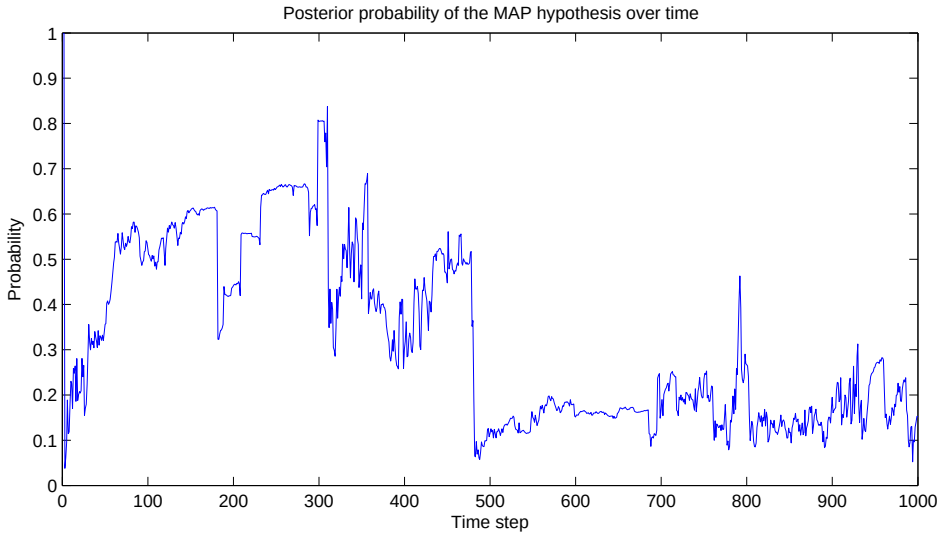

Figure 4.14: The posterior probability of the MAP estimate over time for the run shown in Figure 4.13.

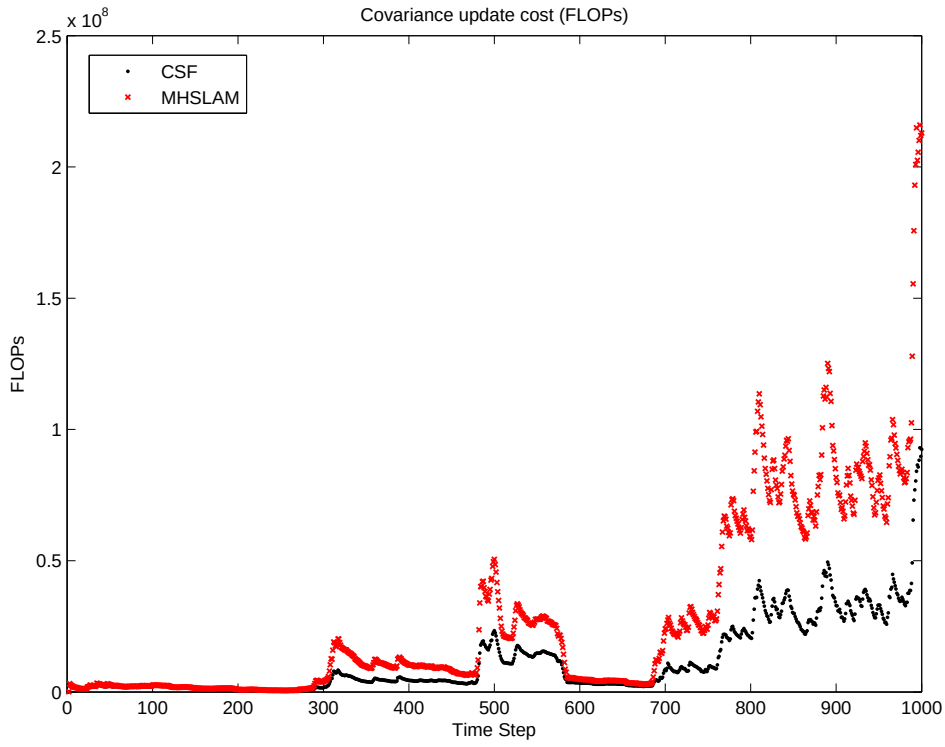

Figure 4.15: Approximate update cost of full MHSLAM and the CSF for all hypotheses. The computational complexity of the CSF is approximately half that of full MHSLAM for this time period, and we would expect the discrepancy to grow as the run progresses and the dimension of the state increases.

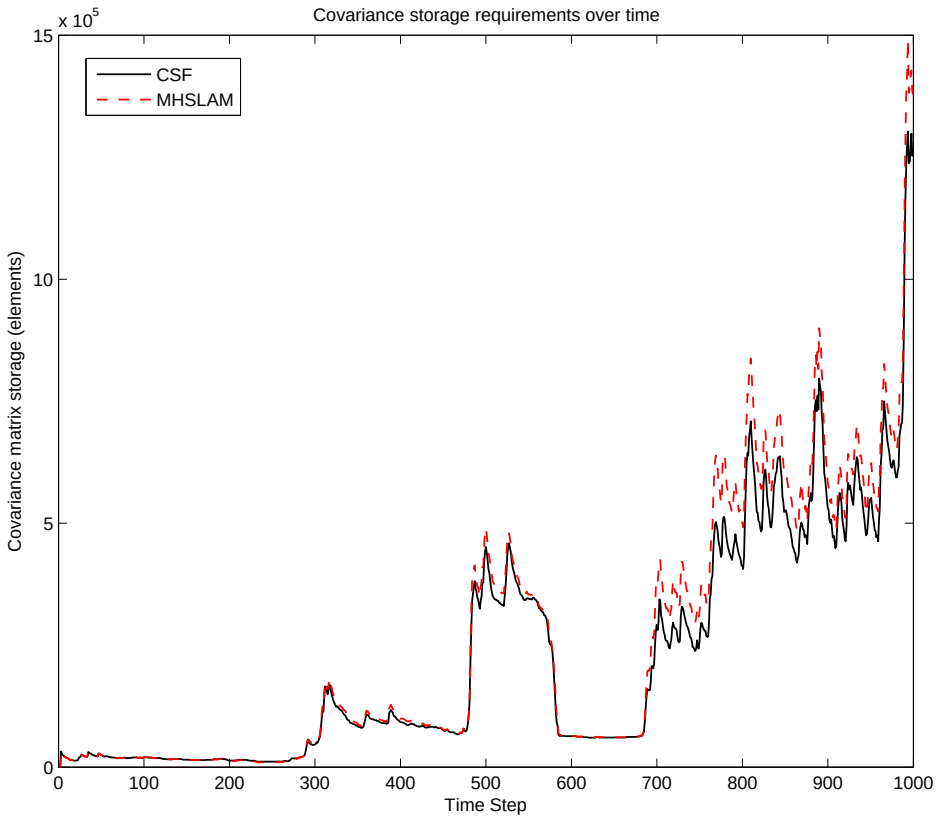

Figure 4.16: Approximate number of elements required to store the covariance matrices for all hypotheses (not exploiting symmetry). Here the CSF sees a slight reduction in the storage requirements compared to full MHSLAM.

order terms and weights are neglected). As the number of beacons in the state increases over time, the difference in update cost between the CSF and full MHSLAM becomes substantial; over double the FLOPs are required for full MHSLAM. At the end of the run the total update cost for full MHSLAM was 27.9 GigaFLOPs, while the CSF required less than half, at 11.6 GigaFLOPs. However regular EKF-SLAM with known relations would have required around 2.2 GigaFLOPs, almost a fifth that of the CSF. The storage saving of the CSF over full MHSLAM is far smaller than the computational saving; this is unsurprising given that the EKF storage cost is $O(n^2)$ while the computational cost is $O(n^3)$. Thus the benefit of the CSF appears to be more in the update cost rather than storage.

We note that both the storage and computational cost could potentially be improved if more intelligent methods than a basic time metric of determining whether to move beacons into the common state were applied.

4.6 Conclusion

In this chapter we considered the problem of estimating the presence of common underlying structure (and thus relations) between features in SLAM and a prior map through the use of MHT, resulting in MHSLAM. We used a prior map accuracy of 0.5 m (1σ) with a 30% clutter level, and were able to exploit about 80% of the relations present (around 40 relations), with an average of only 1.6 incorrect relations applied. Exploiting these improved the quality of SLAM dramatically; the average rMSE improved from 3.25 to 0.45 m, and 80% of the Monte Carlo runs met the 3σconsistency criterion, compared to only 28% for SLAM without a prior map.

To perform MHSLAM more efficiently, we derived the Common State filter, a suboptimal method of performing EKF-based MHSLAM that is more efficient than full MHSLAM. The performance is comparable to full MHSLAM, at a significant saving in update complexity (27.9 vs 11.6 GigaFLOPs in our experiment), and slight saving in storage (1.25 vs 1.38 million elements). In general, the information loss has an almost negligible impact on the platform pose estimate and quality of the map produced. In the worst case the CSF is equivalent to full MHSLAM.

The CSF could also be used in this context of Lazy data association [51], in which case one would not need to consider the common state when revising data associations, resulting in significant computational savings.

However as with full MHSLAM, the CSF still theoretically suffers an exponential computational increase with the number of ambiguous relations. The computational cost will also fluctuate as beacons are moved into the hypothesis states. In addition, there is no lower bound on the computational saving compared to full MHSLAM; it is possible that under certain conditions, such as during loop closing, that all beacons will be moved into the hypothesis states, in which case the cost will be the same as full MHSLAM. This reduces the applicability of the CSF in platforms that require a predictable computational cost without compromising on the accuracy level compared to full MHSLAM.

While effectively able to disambiguate correct hypotheses as well as full MHSLAM, it is still too computationally expensive in our case to provide sufficient performance for evaluating (4.5), suggesting that the strategy of enumerating all the hypotheses and incrementally evaluating (4.8) at each time step

is too computationally expensive for a practical SLAM solution, even if each state benefits from the computational and storage benefits that the CSF provides.

In the next chapter we will trade off the ability to use relation information immediately for computational efficiency, evaluating (4.12) after a batch of observations has been made, thus delaying the application of constraints until we are sure there is a common structure. This only requires a single EKF state, and thus we do not need to use the MHSLAM approach.

Chapter 5

Resolving Common Structure with a Single State

In the previous chapter we presented an undelayed incremental strategy based on multiple hypotheses for incrementally determining structure between features in the SLAM state and the prior map. Using the Common State Filter (CSF), we can do this in a manner that has superior computational performance to full MHSLAM. However, like full MHSLAM, because the CSF does not delay applying common structure information, and the number of hypotheses (and thus states) grows exponentially with the number of ambiguous structures, this combined with the large number of features and observations required to determine the correct relation hypothesis (and thus which structures are common) makes the method unfeasible in all but the sparsest environments.

In this chapter we show how the problem may be evaluated in a retrospective non-MHSLAM framework, by building up parts of the map in the SLAM state before resolving of common structure hypotheses. As an example of the concept behind this, consider the scenario in Figure 5.1, showing a structure $\mathbf{s}_{w\{s\}}$ corresponding to a building wall in the physical world, and two map estimates $p(\mathbf{x}_p|\mathbf{z}_p)$ and $p(\mathbf{x}_m|\mathbf{z}_m)$, the first a map of dense points created by a platform performing SLAM, and the second a map of planes representing building walls, obtained by surveying. From the maps so far we can see evidence of the presence of common underlying structure and its possible type in both maps, without having to speculatively create hypotheses from the observations. This allows us to directly compute the posterior probability of relations holding across hypotheses (as in MHSLAM), but without explicitly enumerating and propagating a state for each hypotheses. Thus we can maintain a single state regardless of the number of potential relations with the prior map.

The trade-off is that we delay making use of relations until we can determine whether the structure is common (and thus whether they apply), and thus lose the EKF error mitigation benefits that applying relations from common structure immediately confers. This is more akin to a batch rather than incremental approach to using the prior information, as we batch up the features in the maps and process them together at once, rather than as observations are received. This also allows us to effectively use information from features in a single map with structure models to inform a prior for a structure being present.

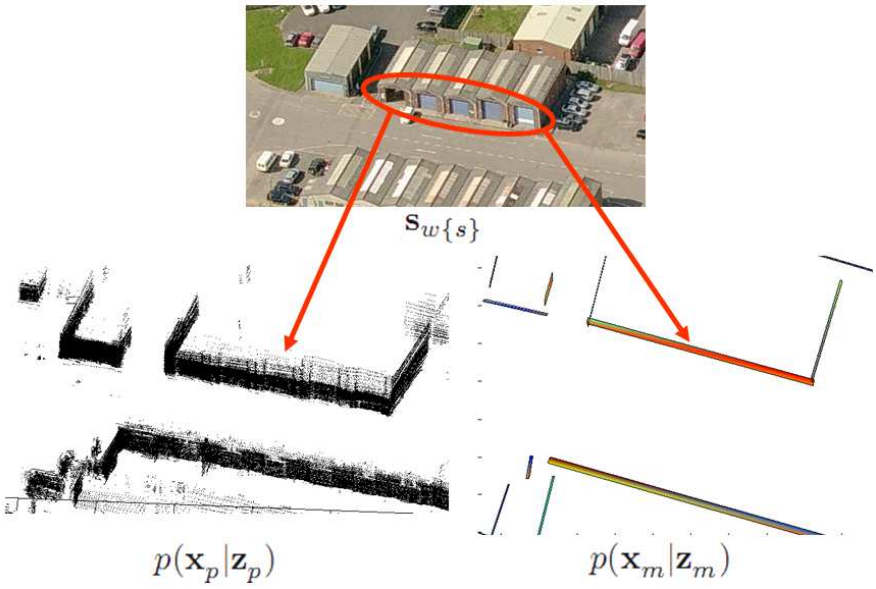

Figure 5.1: Example showing two map estimates $p(\mathbf{x}_p|\mathbf{z}_p)$ and $p(\mathbf{x}_m|\mathbf{z}_m)$, the left consisting of dense point cloud features, and the right of planes. Features that correspond to the same structure instance in the real world (a wall shown at the top) are indicated. This known common structure allows us to relate features between the two maps, and use classifiers to identify which features in each map may have come from this structure type. (Image source: www.multimap.com)

In this chapter we show how we directly implement the framework in chapter 3 in EKF-SLAM using a single state. We then show how information from features in one of the maps can be used to inform the prior on the geometric likelihood for a structure being present. Finally we compare this approach with the incremental CSF approach from the previous chapter, and use it to investigate the effect of spurious features, varying clutter and prior map accuracy on EKF-SLAM.

5.1 Batch Inference

In the previous chapter we showed how $p(d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\})|\mathbf{z}_{p\{i\}}(k_1 : k_a), \mathbf{z}_{m\{n\}})$, the probability of the relation F holding between the ith feature in the SLAM state $\mathbf{x}_p$ and the nth feature in the prior map $\mathbf{x}_m$, given the observations $\mathbf{z}$ may be evaluated incrementally for all the observations from time steps $K = k_1 : k_a$, $\mathbf{z}_{p\{i\}}(K)$. We used a multiple hypothesis approach to evaluate the posterior probability distribution over all hypotheses, thus allowing us to use structure information immediately, discarding improbable hypotheses over time. However maintaining a state for each hypothesis can be too computationally inefficient to be useful.

Fortunately we can delay applying constraint information, evaluating the posterior probability directly from a single state.

To infer the probability of a common structure being present, $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ (which we abbreviate to $d_{in,s} = 1$), note from (3.47) that

$$p(d_{in,s}=1|\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m)=\eta\, p(\theta_{in}=\phi|d_{in,s}=1,\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m)\, p(d_{in,s}=1), \tag{5.1}$$

where in our case

$$p(\theta_{in}=\phi|d_{in,s}=1,\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m)=p(\mathbf{f}(\mathbf{x}_{p\{i\}},\mathbf{x}_{m\{n\}})=0|d_{in,s}=1,\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m) \tag{5.2}$$

because $\theta_{in}=0$, and $\phi=\mathbf{f}(\mathbf{x}_{p\{i\}},\mathbf{x}_{m\{n\}})$ from (3.11).

Then to apply a constraint given that $d_{in,s}=1$, we use (3.50),

$$p(\mathbf{x}_p,\mathbf{x}_m|d_{in,s}=1,\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m)=\eta\, p(\theta_{in}=\phi|d_{in,s}=1,\mathbf{x}_p,\mathbf{x}_m)\, p(\mathbf{x}_p,\mathbf{x}_m|\mathbf{z}_{p\{i\}}(K),\mathbf{z}_m), \tag{5.3}$$

where $p(\theta_{in}=\phi|d_{in,s}=1,\mathbf{x}_p,\mathbf{x}_m)=p(\mathbf{f}(\mathbf{x}_{p\{i\}},\mathbf{x}_{m\{n\}})=0|d_{in,s}=1,\mathbf{x}_p,\mathbf{x}_m)$.

Thus for a given time step k_a we can compute the probability of any relation hypothesis from a single posterior distribution $p(\mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(K))$, and thus unlike the MHSLAM solution in the previous chapter, we need only maintain a single state. We implement this for EKF-SLAM as follows.

For clarity, let us consider the initial case (before any beacons relate with the prior map) where the regular joint vehicle/map SLAM state represents $p(\mathbf{x}_v(k_a),\mathbf{x}_p|\mathbf{z}_p(K))$, though this could be conditioned on other structures being present. At time k_a, gating (which is equivalent to only considering features where (3.48) will be significant) shows that it is plausible that $d(\mathbf{x}_{p\{I\}},\mathbf{x}_{m\{N\}},\{\mathbf{s}_{w\{S\}}\})=\mathbf{1}$, or that the set of point beacons I in the state, $\mathbf{x}_{p\{I\}}$ potentially relate with a set of prior map line segment features N, $\mathbf{x}_{m\{N\}}$ if both came from the flat wall structure instances S, $\mathbf{s}_{w\{S\}}$. Thus we have a vector of indicators which evaluate to a binary vector C, $d(\mathbf{x}_{p\{I\}},\mathbf{x}_{m\{N\}},\{\mathbf{s}_{w\{S\}}\})=C$, where for the jth feature pair $C^j=d(\mathbf{x}_{p\{I^j\}},\mathbf{x}_{m\{N^j\}},\{\mathbf{s}_{w\{S^j\}}\})$, the superscript indicating an index within each set. For clarity we can abbreviate $d(\mathbf{x}_{p\{I^j\}},\mathbf{x}_{m\{N^j\}},\{\mathbf{s}_{w\{S^j\}}\})$ to $d^j_{IN,S}$. We wish to find the maximum likelihood estimate for C, C_{ML}. In our EKF-SLAM based implementation we cannot revise incorrectly applied relations[1] and thus we additionally want to ensure that C_{ML} is significantly more probable than the next most probable value.

We can do so by enumerating all the combinations that C can take to obtain the set of hypotheses $C_1 \ldots C_N$, and then computing the probability distribution $p(d_{IN,S}|\mathbf{z}_p(K),\mathbf{z}_m)$ over these hypotheses from (5.1). The likelihood function we use is given by (4.17).

We consider hypotheses with a probability below a lower threshold p_{reject} (such as 0.02) to be false; the probabilities of these are set to zero and the remaining values renormalised to give the posterior probabilities of the remaining hypotheses (this parallels the pruning stage in MHSLAM).

However, unlike MHSLAM we cannot track the states for all these hypotheses (using the MAP hypothesis as the representative state), because we only maintain a single state. This state represents the joint vehicle map/estimate $p(\mathbf{x}_v(k_a),\mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(K))$ where the ambiguous relation $d_{IN,S}$ is marginalised out, and is thus conservative. We either need to establish that one of the hypotheses is correct and commit to it (updating our state to reflect it), or commit to none at this time, and maintain the current state. Optionally if there are few hypotheses at this point we could choose to spawn and track

[1] However reversible association is the subject of future work.

Table 5.1: Example showing the posterior probabilities associated with four hypotheses created from two ambiguous structure relations. Although none of these hypotheses exceeds the threshold $p_{accept} = 0.95$, marginalising out $d_{in,s}$ gives two remaining hypothesis, one of which exceeds the acceptance threshold. Thus summing over the hypotheses of $d_{in,s}$ gives $p(d_{jn,s} = 1) = 0.5 + 0.451 = 0.951$.

Hypothesis:	1	2	3	4
$d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = d_{in,s} =$	1	1	0	0
$d(\mathbf{x}_{p\{j\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = d_{jn,s} =$	1	0	1	0
Probability:	0.451	0.028	0.5	0.021

states for the hypotheses using the CSF. In this case the current state would be duplicated to as many hypotheses as there are, and the state for each hypothesis would be updated according to which relations are hypothesised to apply, as in MHSLAM.

We accept a hypothesis h as being correct (and thus the others as being false) if its probability exceeds a probable upper threshold value p_{accept}, such as 95%. If no hypotheses meet the threshold, we may still be able to form a hypothesis that exceeds p_{accept} by marginalising over the relation hypotheses of some beacons as shown in Table 5.1; the relation status of these beacons will be considered ambiguous at this time (however can still be resolved at a future time step). Naturally this marginalisation strategy is only applied when there are multiple ambiguous relations.

Unlike MHSLAM, which commits to the set of hypotheses immediately and thus must compute the posterior probabilities using (4.5) after every observation, (5.1) does not need to be computed after every update, rather every $\triangle k$ time steps depending on computational considerations.

The choice in posterior threshold is a trade-off between the performance (accuracy and consistency) impact of being either conservative or optimistic when determining the structure indicator status. Being optimistic with a low acceptance threshold p_{accept} carries a performance penalty from false positives (applying a relation when the structure is not common). However being conservative also means that more time elapses before a relation is applied, giving a less optimal estimate and allowing time for EKF errors to grow. We use a p_{accept} threshold of 70% in our simulations as this gave the best results.

If a hypothesis is accepted, the state is conditioned on the outcome of the relations. Thus if the accepted hypothesis is that $d_{in,s} = 1$, the state is updated based on the relation this entails, according to section 3.6 in chapter 3.

5.2 Informing a Structure Prior

Consider again the scenario in Figure 5.1. In this case a relation could be formulated for the ith point and nth plane that were generated from the same wall s, which we indicate by $d(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1$ (also shown as $d_{in,s} = 1$ for brevity).

In section 3.7 we mentioned that $p(d_{in,s} = 1)$ is a prior to (3.47), which infers whether $d_{in,s} = 1$ conditioned on the geometric compatibility of the estimates of $\mathbf{x}_{p\{i\}}$ and $\mathbf{x}_{m\{n\}}$. For a large number of features (3.47) can be expensive to compute, so if we make the prior as informative as possible we can

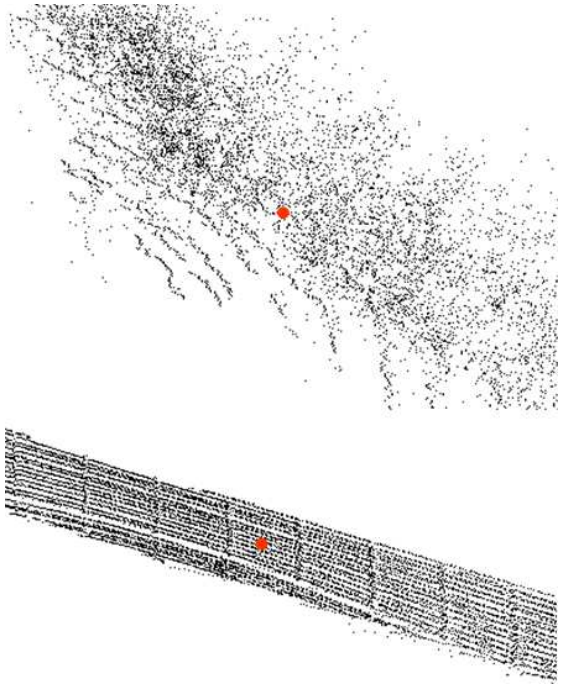

Figure 5.2: Comparison of two point clouds in the map $\mathbf{x}_p$. The large points in the centre of each show the possible location of the ith point $\mathbf{x}_{p\{i\}}$. By considering the structure of the surrounding points, we can see that the top point cloud $\mathbf{x}_{p\{i\}}$ is unlikely to correspond be a wall structure $\mathbf{s}_w$, thus a classifier would give a low value to $p(d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_w\}) = 1|\mathbf{x}_p)$. Conversely, the bottom point cloud is wall-like, and thus there is a high probability that $\mathbf{x}_{p\{i\}}$ came from a wall.

reduce the number of features we have to consider. We do this by conditioning the prior on *classifiers*, or models of how a given structure appears in a feature map produced under a particular map model, giving it the form $p(d_{in,s} = 1|\mathbf{z})$. Thus for the scenario above, we could infer the probability that each point in the map $\mathbf{x}_p$ was generated by a wall $\mathbf{s}_w$, and thus reject the majority of points that are not "wall-like" before comparing features between the maps in (3.47), improving computational performance and reducing false positives.

It is useful to look for structure a priori in this way for computational reasons when dealing with dense maps and for determining which structures are present. There are a number of classifiers that could be used for discovering and recognising structure in the environment [99, 76, 2]. As discussed in chapter 3 these classifiers have yet to be exploited in this way.

The prior in (3.47) can be broken down into a probability for each map, for instance $p(d(\mathbf{x}_p, \{\mathbf{s}_{w\{s\}}\}))$ corresponding to the feature map $\mathbf{x}_p$ and structure instance $\mathbf{s}_{w\{s\}}$ according to (3.5); for brevity we can denote $d_{\cdot,s} = d(\mathbf{x}_p, \{\mathbf{s}_{w\{s\}}\})$, the dot signifying any instance. Let us consider the feature map of points $\mathbf{x}_p$ and the structure instance $\mathbf{s}_{w\{s\}}$ representing a flat wall. A classifier, which models the configuration of points produced from flat walls and other structures can be used to estimate the probability that a point $\mathbf{x}_{p\{i\}}$ was generated by a flat wall $\mathbf{s}_w$ based on this point and others in $\mathbf{x}_p$, as shown in Figure 5.2.

Probabilistically this is represented as $p(d_{i,\cdot} = 1|\mathbf{x}_p)$ (where $d_{i,\cdot} = d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_w\})$ for brevity), and can be computed from Bayes rule,

$$p(d_{i,\cdot} = 1|\mathbf{x}_p) = \eta\, p(\mathbf{x}_p|d_{i,\cdot} = 1)\, p(d_{i,\cdot} = 1), \tag{5.4}$$

where η is a normalising term and $p(\mathbf{x}_p|d_{i,\cdot} = 1)$ is a likelihood model for the points based on the structure. Because we do not have $\mathbf{x}_p$ but an estimate $p(\mathbf{x}_p|\mathbf{z}_p)$ based on observations $\mathbf{z}_p = \mathbf{z}_p(k_1 : k_a)$, we marginalise out $\mathbf{x}_p$ to condition on $\mathbf{z}_p$,

$$p(d_{i,\cdot} = 1|\mathbf{z}_p) = \int p(d_{i,\cdot} = 1|\mathbf{x}_p)\, p(\mathbf{x}_p|\mathbf{z}_p)\, \mathrm{d}\mathbf{x}_p. \tag{5.5}$$

We then compute the probability that $\mathbf{x}_{p\{i\}}$ is associated with the structure instance s given it came from any instance of $\mathbf{s}_w$,

$$p(d_{i,s} = 1|\mathbf{z}_p) = p(d_{i,s} = 1|d_{i,\cdot} = 1)\, p(d_{i,\cdot} = 1|\mathbf{z}_p), \tag{5.6}$$

noting that because of the structure hierarchy $p(d_{i,s} = 1|d_{i,\cdot} = 0) = 0$. By gating we can usually restrict the candidate structure instances for s to a single potential instance. Under certain circumstances the classifier may specifically detect the structure instance s, in which case it returns $p(d_{i,s} = 1|\mathbf{z}_p)$ directly, and (5.6) is not needed.

Assuming the same for the map $\mathbf{x}_m$ and substituting into (3.5) gives

$$p(d_{in,s} = 1|\mathbf{z}_p, \mathbf{z}_m) = p(d_{i,s} = 1|\mathbf{z}_p)\, p(d_{n,s} = 1|\mathbf{z}_m), \tag{5.7}$$

where $d_{i,s} = d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\})$ and $d_{n,s} = d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\})$.

Note that here we do not use any information as to whether the maps are geometrically compatible and thus obey the relation $\theta_{in} = \phi$ in (3.47), as this is part of the likelihood function $p(\theta_{in} = \phi|d_{in,s} = 1, \mathbf{z}_p, \mathbf{z}_m)$ in (3.47). Note also that this means that the two probability distributions are independent (assuming there is no other source of dependency a priori between the maps).

Thus inferring the presence of common structure between features in maps does not solely depend on evaluating the geometric likelihood between the maps in (3.48). In situations where computation is limited or there is high positional uncertainty of features in the maps, such prior classifier information is likely to be very useful.

For the simulations below, the sparsity of the environment is such that we do not require an explicit classifier, instead concentrating on the performance of the geometric likelihood at disambiguating clutter with a constant prior. However in chapter 6 where we perform a real experiment we use a classifier (with a simple heuristically-derived likelihood) to determine whether features are likely part of a wall structure.

5.3 Simulations

5.3.1 Comparison with CSF-MHSLAM

We use the same simulation scenario as used in chapter 4 to compare the performance of the single state method with CSF based MHSLAM.

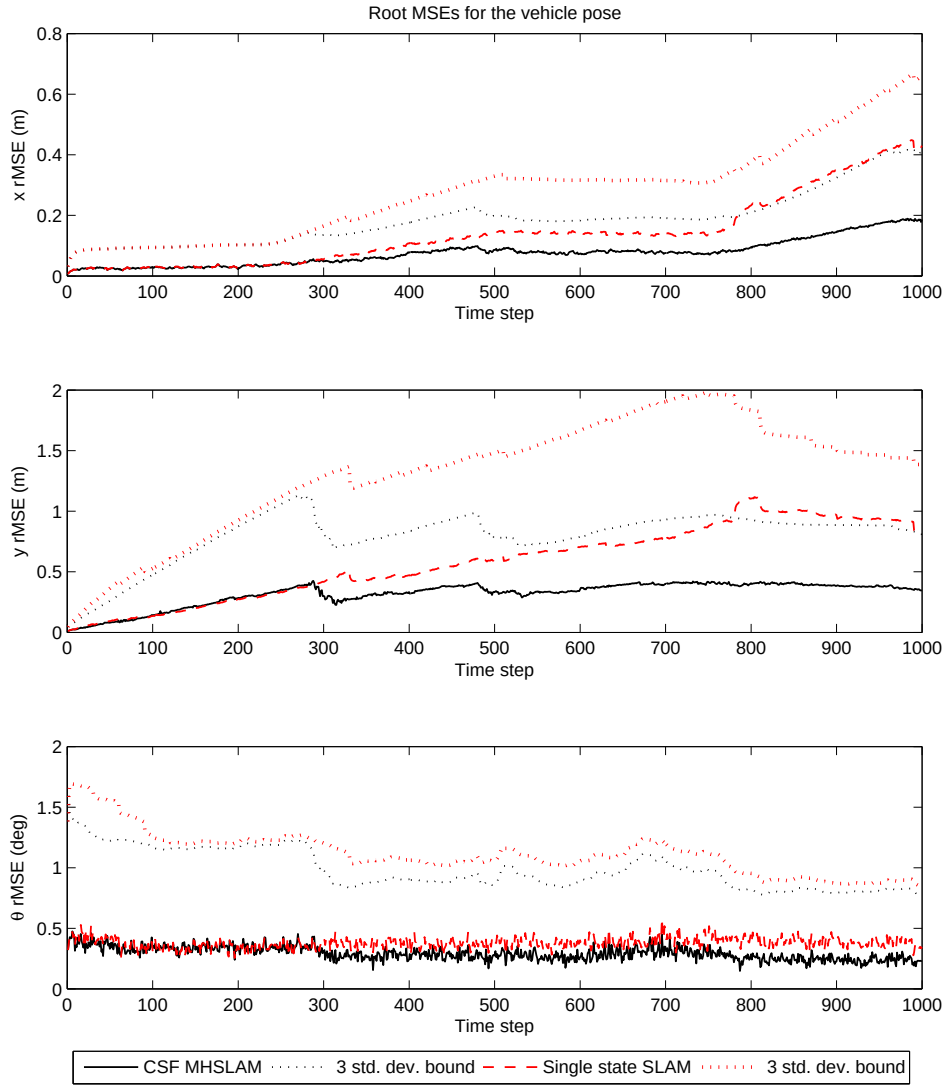

Figure 5.3: rMSE comparison between the CSF MHSLAM from the previous chapter and the single state method in this chapter for resolving relations. The single state method has produced a less optimal estimate, however the errors of both methods lie comfortably within their 3σ uncertainty estimates.

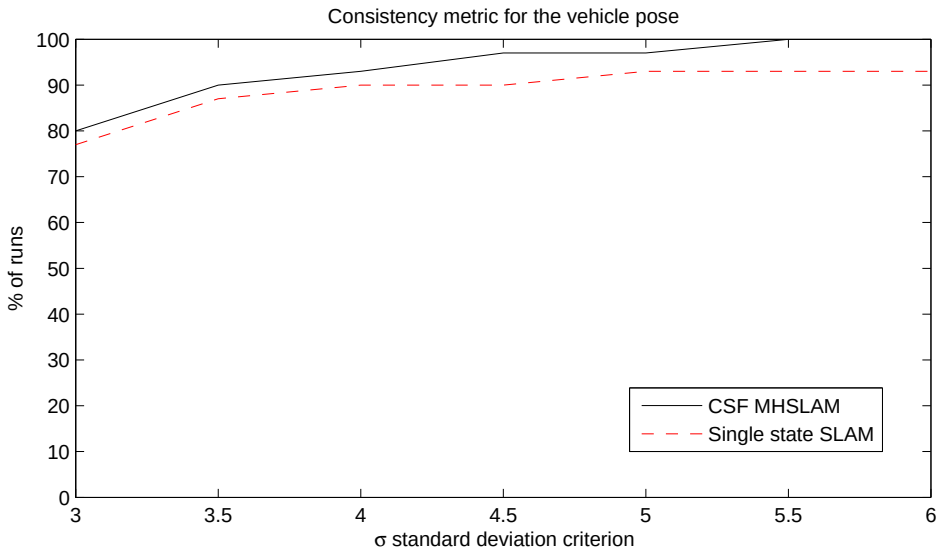

Figure 5.4: Inconsistency severity comparison between the CSF and single state method. The consistency metric indicates for what percentage of runs the vehicle pose was within a given standard deviation criterion for at least 95% of the run. Both methods exhibit a similar, low degree of inconsistency.

Figure 5.3 shows that the single state method is suboptimal compared to the CSF estimate, as the uncertainty and error is greater. The CSF estimate also has slightly better consistency as shown by Figure 5.4. This is because the CSF can apply relations immediately without delay, whereas the single state method has to wait until the posterior probability of a set of relations holding exceeds the threshold. For the CSF results we show the MAP hypothesis at any time step, which is likely to contain incorrect relations if it is not the correct hypothesis. However despite this, it appears to give better performance because the greater number of relations applied offsets the negative effect of unmitigated linearisation and EKF errors.

Although in this case the CSF has a more accurate vehicle estimate, the number of relations applied (both correct and incorrect) is proportionally higher than that of the single state method, as shown by Figure 5.5. This suggests that the single state method has been more conservative, which although resulted in less incorrect relations being applied, has given a significantly less accurate vehicle pose estimate and slightly lower consistency. This suggests that in our case applying relations optimistically (with a greater number of incorrect relations) gives better results than being conservative.

It appears that the effect of the nonlinear system or EKF errors (or both) on the likelihood computation is to make the system more optimistic in the case of immediate relation application (CSF), and relatively more conservative in the case of delayed relation application (single state method). However both strategies perform similarly in terms of the number of true compared to false positives, indicating similar relation resolving ability.

The computational penalty the CSF sees for the more accurate vehicle pose estimate is shown in Figure 5.6, which shows the storage cost, and Figure 5.7 which shows the covariance update cost for all the hypotheses.

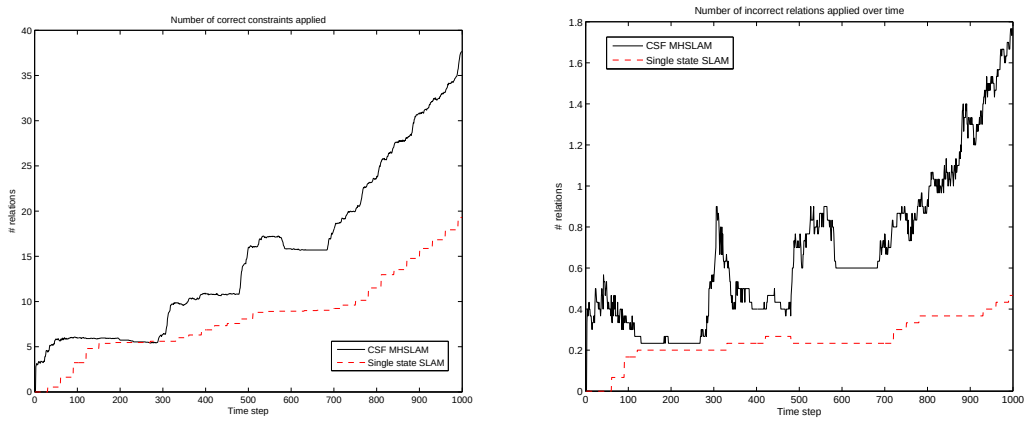

Figure 5.5: Left: Comparison of the number of correct relations present in the state at each time step for the single state and multiple hypothesis methods. Right: Comparison of the average number (over 30 MC runs) of incorrect relations present in the state at each time step for the single state and multiple hypothesis methods. In the MHSLAM case this corresponds to the most likely hypothesis at each time step, and thus the number can decrease when this hypothesis changes.

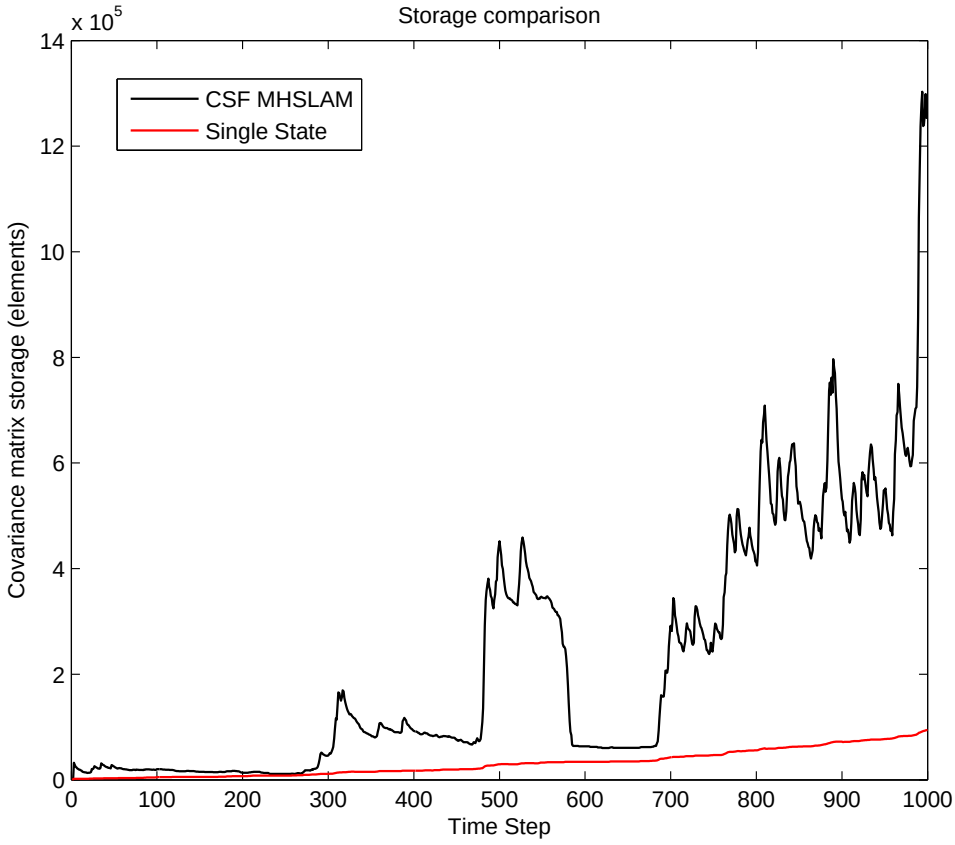

Figure 5.6: Covariance storage cost (not exploiting symmetry) for the single state method and the CSF.

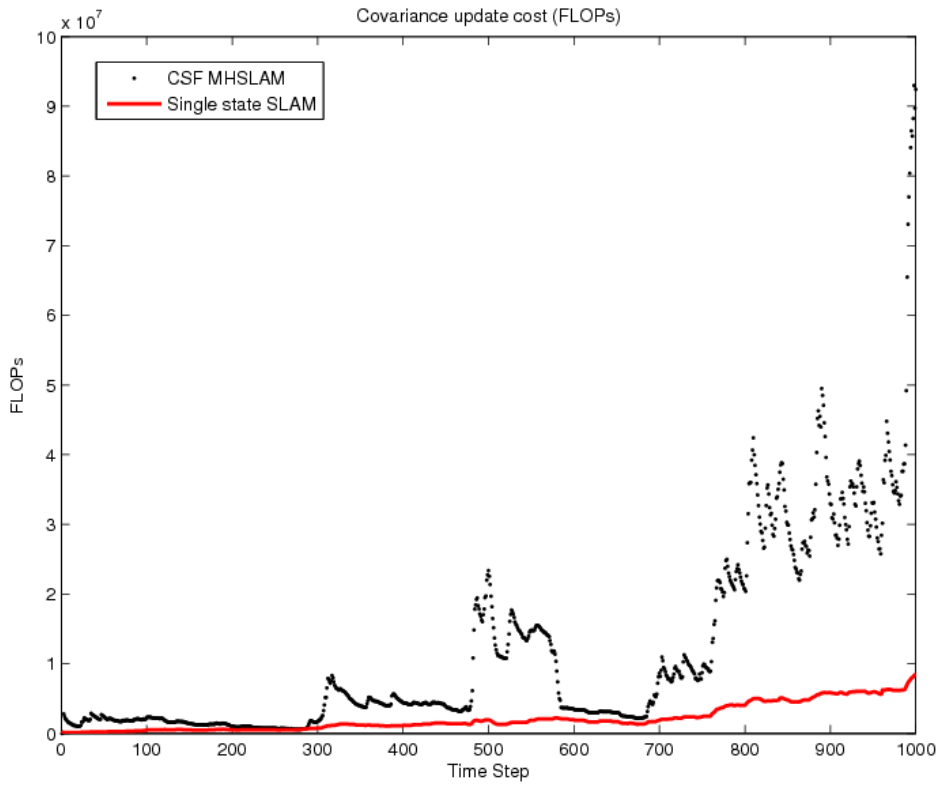

Figure 5.7: Computational cost of updating the covariance of all the hypotheses (not exploiting symmetry), for the single state method and CSF.

At the end of the run the single state method had performed 2.22 GigaFLOPs for the covariance update, while the CSF required 11.6 GigaFLOPs, a five-fold increase. Unlike the CSF, the complexity of the Kalman update for the single state method is independent of the number of potential relations, and thus the method retains the computational update complexity equivalent to EFK-SLAM with known relations.

Since the relation resolution performance of the CSF approach does not appear to be better than that of the single state approach, while the computational costs are far greater, for investigating the effect of clutter and prior map accuracy we use the single state method. As in chapters 2 and 3 for computational reasons (we do not use submapping for instance) we discard beacons not seen after 5 s for beacons not associated with any structure, with the exception of beacons in the loop closure region. Beacons associated with structure are only discarded after the last beacon that gated with their line segment has not been observed for 8 s. The vehicle travels at 14 m/s, thus 8 s corresponds to a distance of up to 112 m behind the vehicle.

5.3.2 Robustness to Spurious Line Segments

In this scenario the prior map has a 1σ error of 1 m, and 6 spurious line segments have been added to the 48 already present in the prior map. Thus 11% of the line segments are spurious; that is they do not exist in the real world so are not associated with any structure. This could occur because of a mapping errors or because their structures are no longer present in the world. Thus with reference to section 4.5.2, $p(d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1) = 0.89$, that is the nth line segment has an 89% chance of being generated by a wall in the real world. The clutter density of points (the proportion that were not generated by the same wall as the line segment they gated with) is 40%, thus $p(d(\mathbf{x}_{p\{i\}}, \{\mathbf{s}_{w\{s\}}\}) = 1 | d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1) = 0.6$.

Figure 5.8 shows the absolute error in the robot pose estimate, along with the 3σ bound of the error estimate. The SLAM system using the prior map shows a dramatic improvement in consistency and accuracy from the prior information, showing that even with the high clutter level and spurious lines, the geometric prior map significantly improves SLAM, even when the 1σ error in the prior map is 1 m (compared with the 0.01 m range accuracy of the sensor).

Figure 5.9 shows that the number of constraints applied where there was common structure between the features (and thus the constraint holds). For most of the trajectory no constraints were applied incorrectly (as indicated by the missing values), however for the few that were, the number applied correctly was between one and two orders of magnitude higher on average, showing that the method is effectively able to determine when there is common structure, and discount relation hypotheses for spurious line segments.

In the following sections we consider only the effect of varying the clutter density and prior map accuracy on performance and thus there are no spurious lines in the prior map, thus $p(d(\mathbf{x}_{m\{n\}}, \{\mathbf{s}_{w\{s\}}\}) = 1) = 1$.

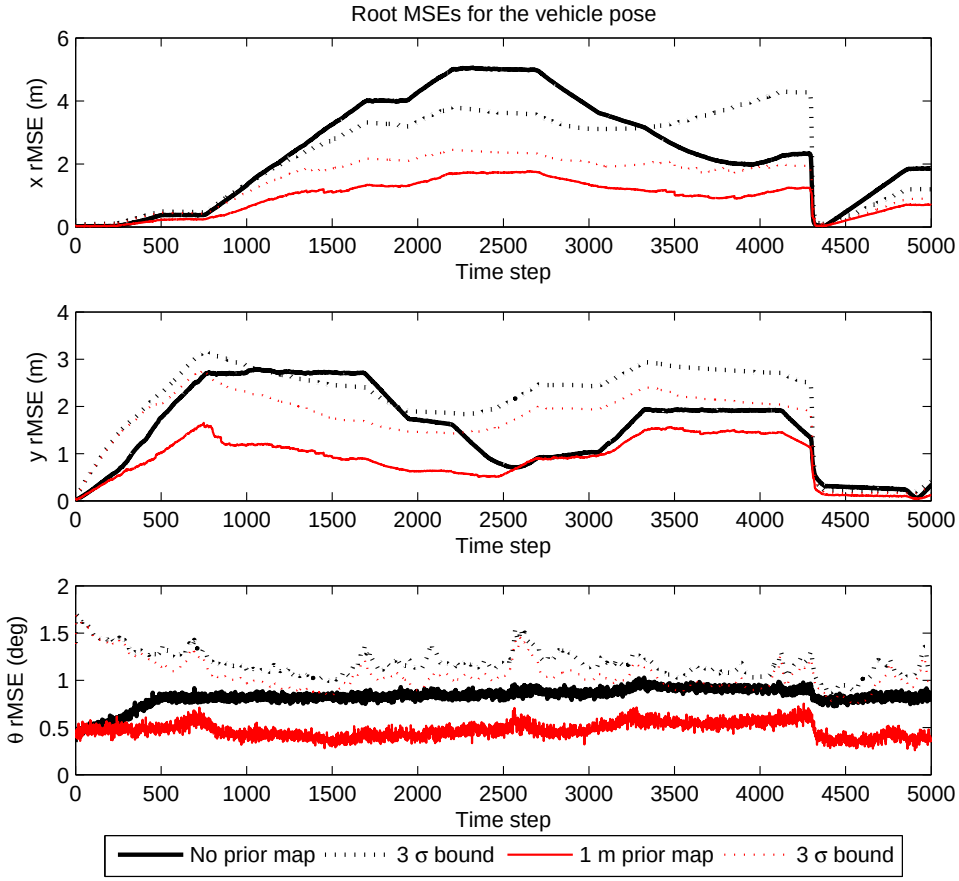

Figure 5.8: Vehicle x, y and θ pose error over the trajectory for regular SLAM without a prior map, and with a prior map of 1 m std. dev. error where 40% of the line segments in the prior map are spurious.

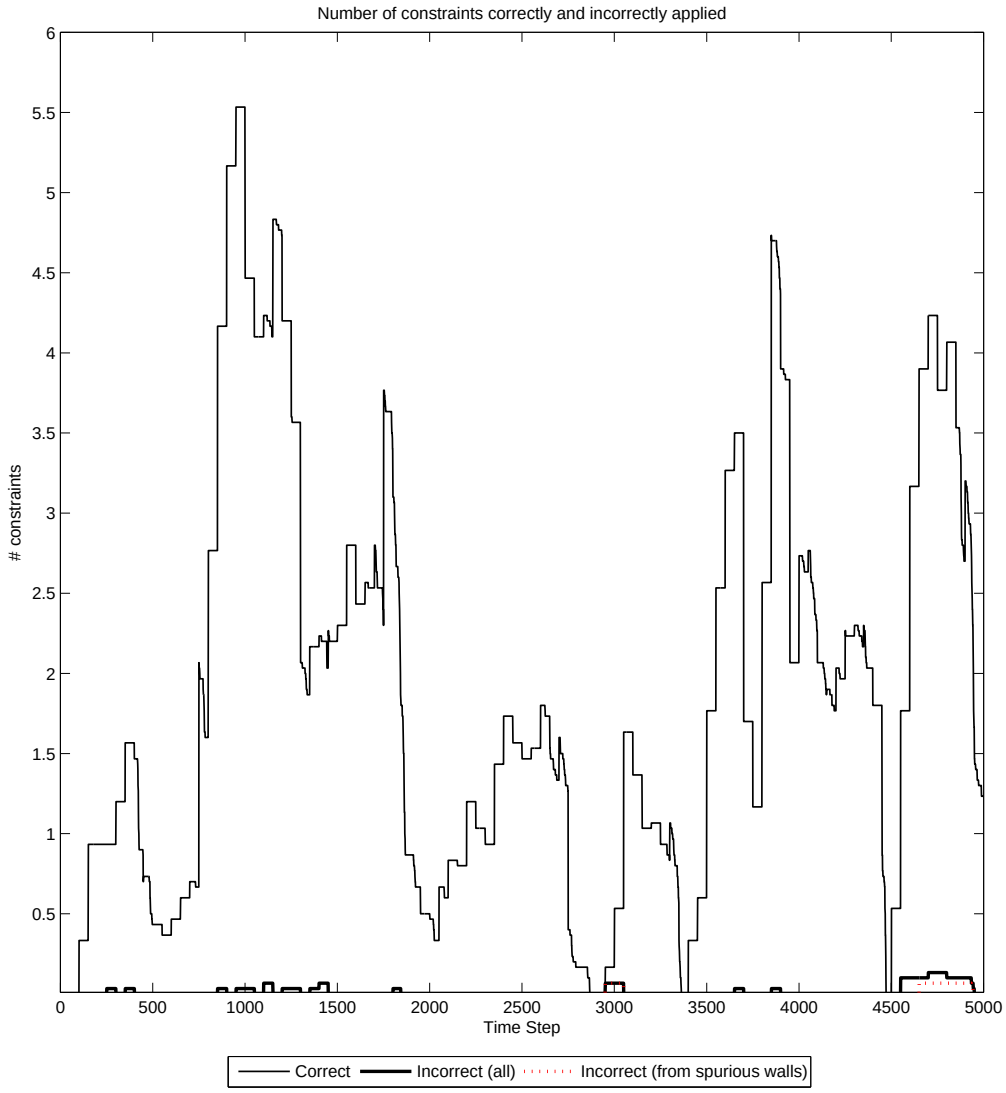

Figure 5.9: Average number of relations applied correctly (thin line) and incorrectly (thick line) to features in the state, with the dotted line indicating how many were from spurious line segments. Because we use a sliding window features are constantly being added and removed from the state. Note how for most of the runs at most 1 or 2 out of the 100 Monte Carlo runs contained any incorrect relations, and thus the average comes to a small fraction below 1.

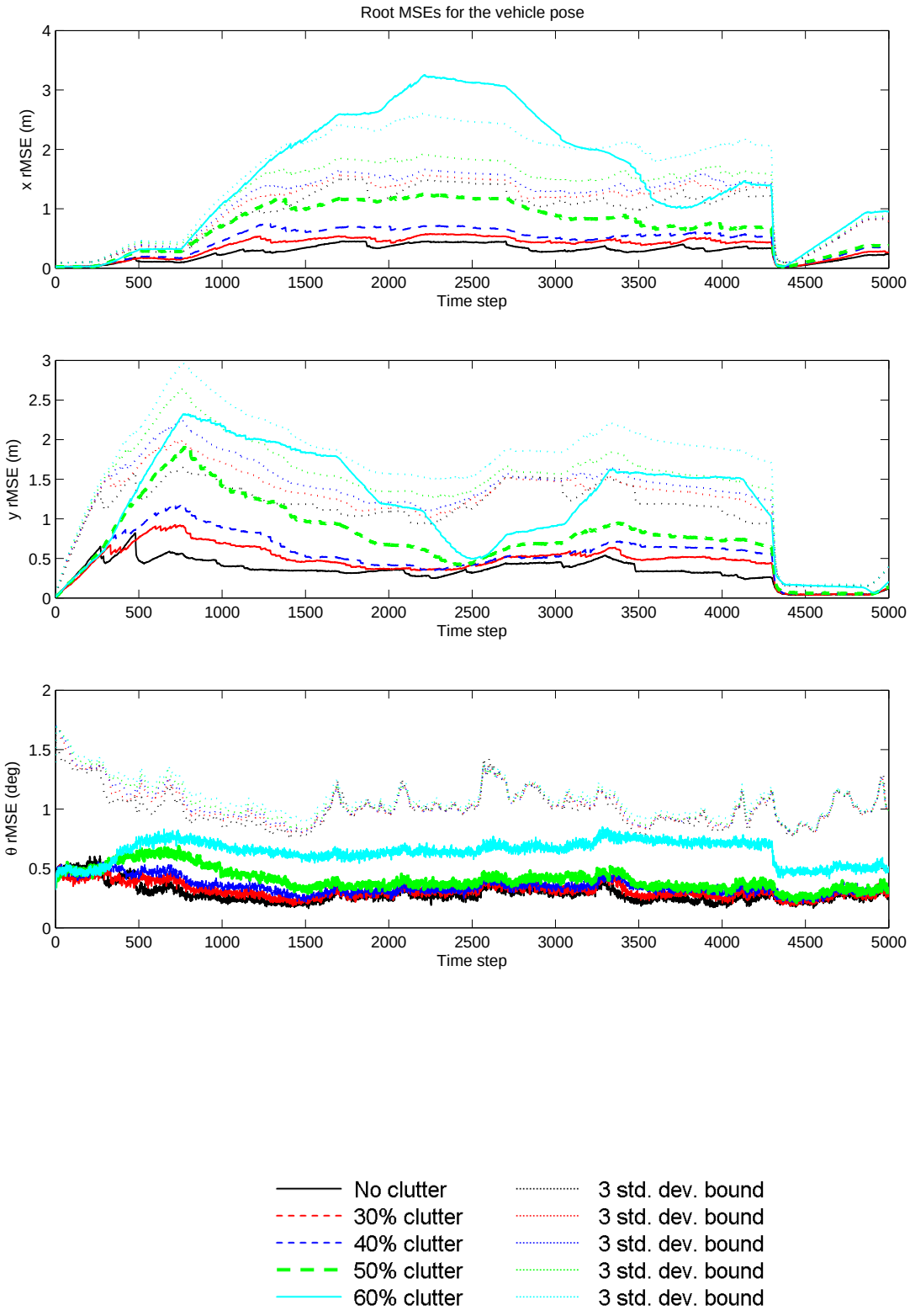

Figure 5.10: Root mean squared error (rMSE) of the vehicle poses with four clutter densities and the case with known relations (i.e. no clutter) shown for comparison. From top to bottom x, y and θ. 3σ standard deviations are shown as dotted lines. Loop closure occurs at 4310 s. The average vehicle pose x, y and θ elements remain within their average 3σ bounds in all cases.

5.3.3 Varying Clutter Density with a Constant Prior Map Accuracy

Figure 5.10 shows the effect of increasing the clutter density on the vehicle pose for a 1 m std. dev. error prior map. The error and uncertainty increase nonlinearly with the degree of clutter, as does the inconsistency. The error and inconsistency increase from going from 50% to 60% clutter is very high; the vehicle x position error is more than double at 2500 time steps, and in the 60% clutter case has exceeded the 3σ uncertainty bound.

Comparison with the ideal case, when the relations are known and thus there is no clutter shows that even at 30% clutter there is a noticeable error increase, however for the prior map accuracy level and sparsity of this environment (average of 1 beacon per 5 m of wall), performance begins to severely degrade when more than 40% of these beacons are clutter (do not lie on the wall).

The orientation is only significantly affected by the degree of clutter for the 60% clutter case, suggesting that for our motion model (that of a steered bicycle), clutter has a modest effect on orientation error compared to position. This is also reflected in the orientation uncertainty estimate, which remains very similar for all the clutter levels considered.

Figure 5.11 gives an indication of the severity of inconsistency in the vehicle pose over all the runs, and thus the probability that any run will remain within a given consistency criterion. Considering the 3σ consistency bound, for unconstrained SLAM less than 30% of the runs meet this criterion. With a sparse high clutter (60%) environment, this increases to 48%. Without clutter this becomes 87%.

We see the nonlinear increase in inconsistency at the 6σ bound; for the 60% clutter case, only 80% of the runs meet the criterion, whereas the clutter levels of 50% and below exceed 94%. All clutter levels see significantly more successful runs compared to unconstrained SLAM, where 63% of the runs exceed the criterion. This shows that all the clutter levels considered reduced the inconsistency in EKF-SLAM, the degree being dependent on the degree of clutter.

Comparison with the ideal no clutter case (where the relations are known) shows that at 4σ and above the consistency of the 30 and 40% clutter cases approach that of the ideal case, showing that at a relatively low degree of clutter the severity of the inconsistency of EKF-SLAM is significantly reduced, and thus a run at these clutter levels will likely produce a map whose error estimate is representative of the actual error.

Figure 5.12 shows how the degree of inconsistency is negatively correlated with the number of correct relations applied. This suggests that a greater error in vehicle position corresponds to a greater beacon error and thus a lower likelihood of relations being applied. This trend is not seen in the number of incorrect relations applied, suggesting that the incorrect relations do not significantly degrade consistency.

Figure 5.13 shows that the number of beacons with correct relations applied (the beacons actually lie on the wall) in the state at each time step decreases dramatically even for low clutter densities (such as 20%), yet as Figure 5.10 shows this has only a small effect on the vehicle variances. This suggests that applying relations to a small number of beacons may be sufficient to have a significant effect on the performance of the algorithm. The effect of the beacon density (with no clutter) on the vehicle

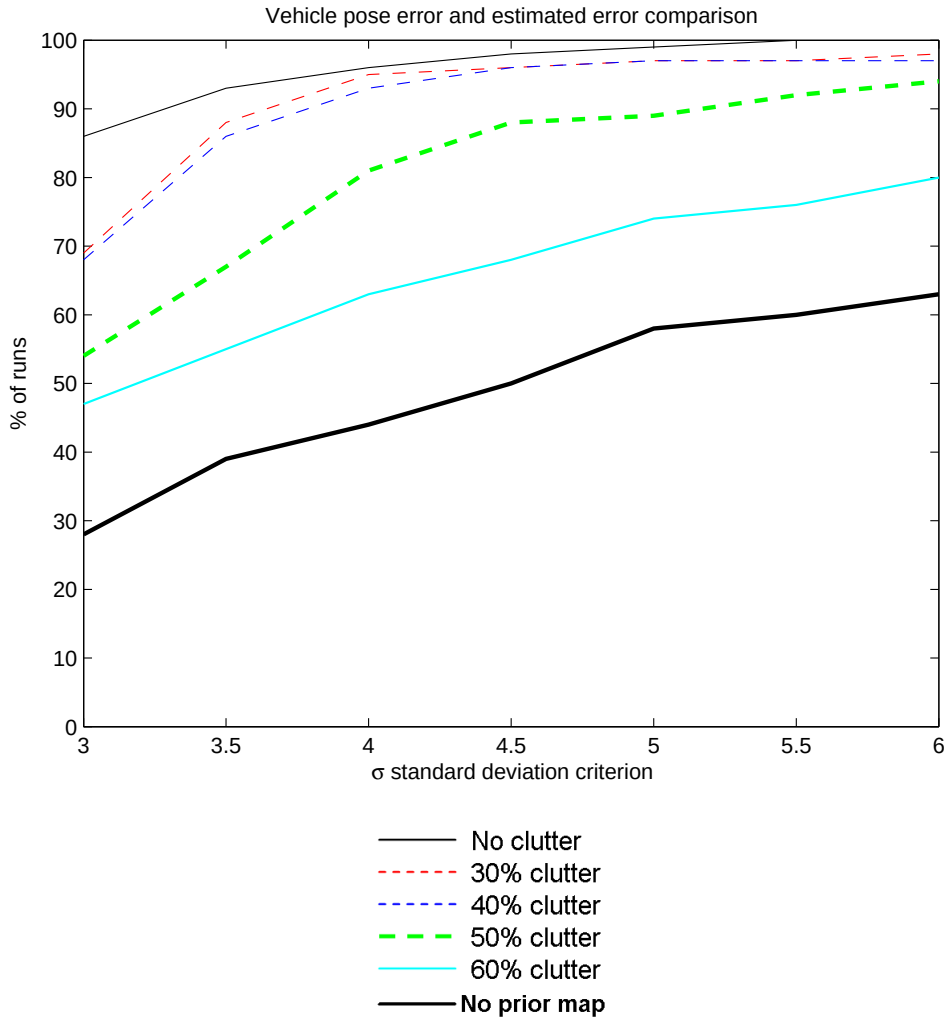

Figure 5.11: Comparison of the number of Monte Carlo runs where the vehicle pose remained within the given number of standard deviations (on the x-axis) for at least 95% of the run. For comparison the case without relation ambiguity (no clutter) is also shown.

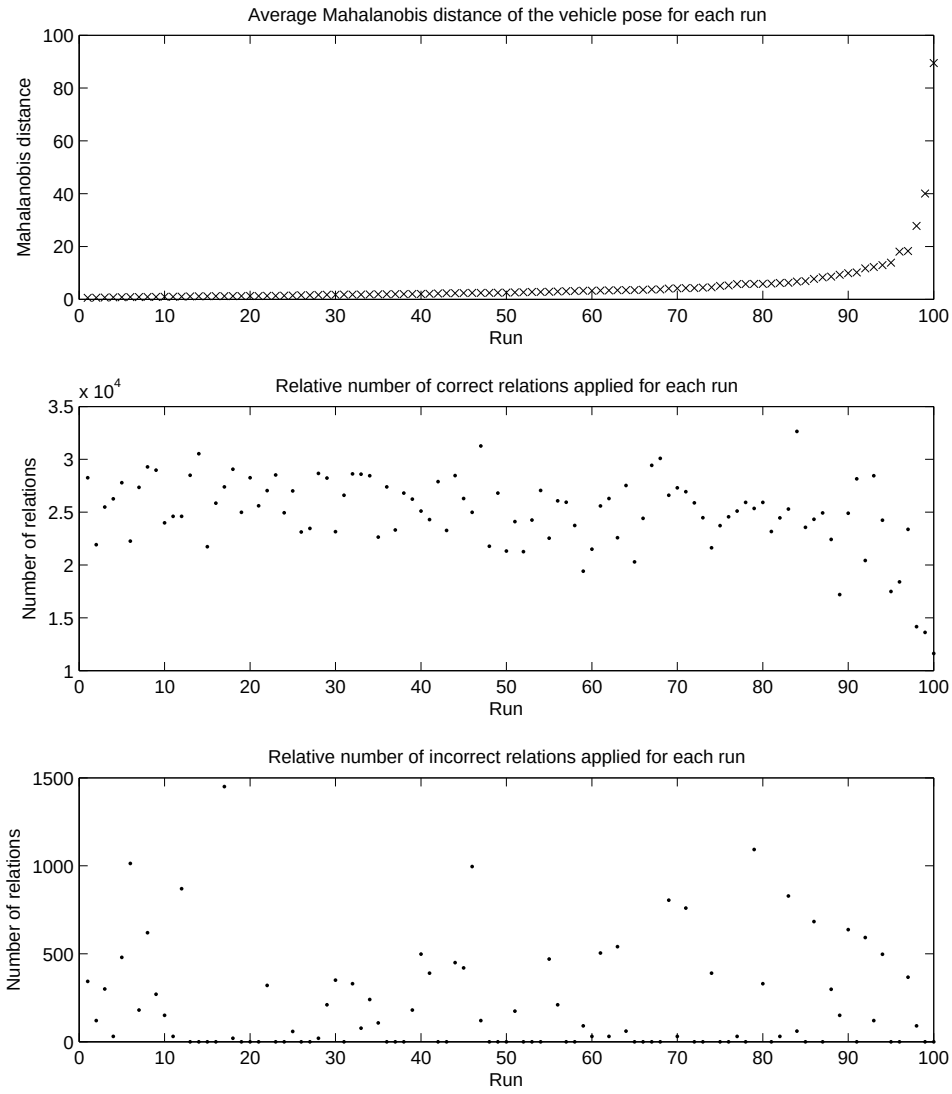

Figure 5.12: Comparison of the number of relations applied for a run with the (sorted) average Mahalanobis distance of the vehicle pose, a higher average Mahalanobis distance indicating worse performance. The average Mahalanobis distance appears to be inversely correlated with the number of relations applied, but uncorrelated with the number of incorrectly applied relations.

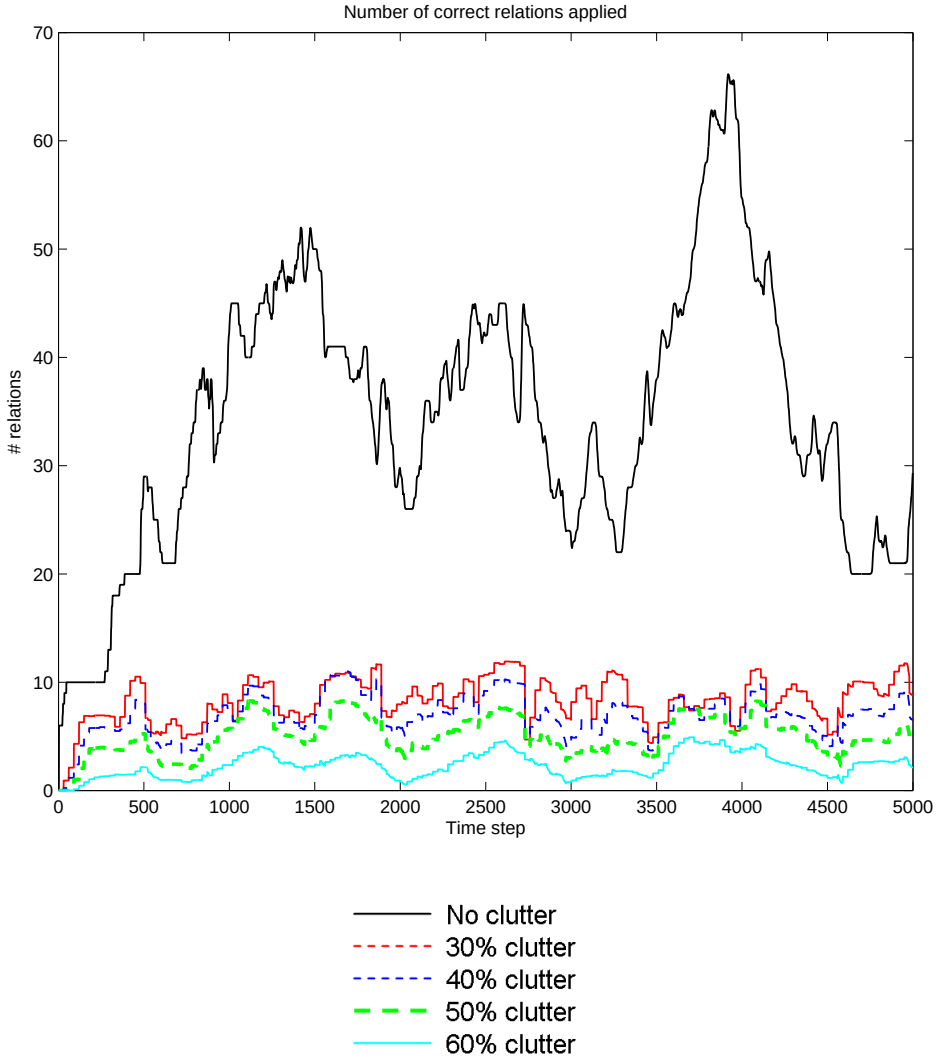

Figure 5.13: Number of beacons with correctly applied relations (they actually lie on their line segment) in the state over time. The case without relation ambiguity (no clutter) indicates the maximum possible number of such beacons (beacons are constantly being removed by the sliding window).

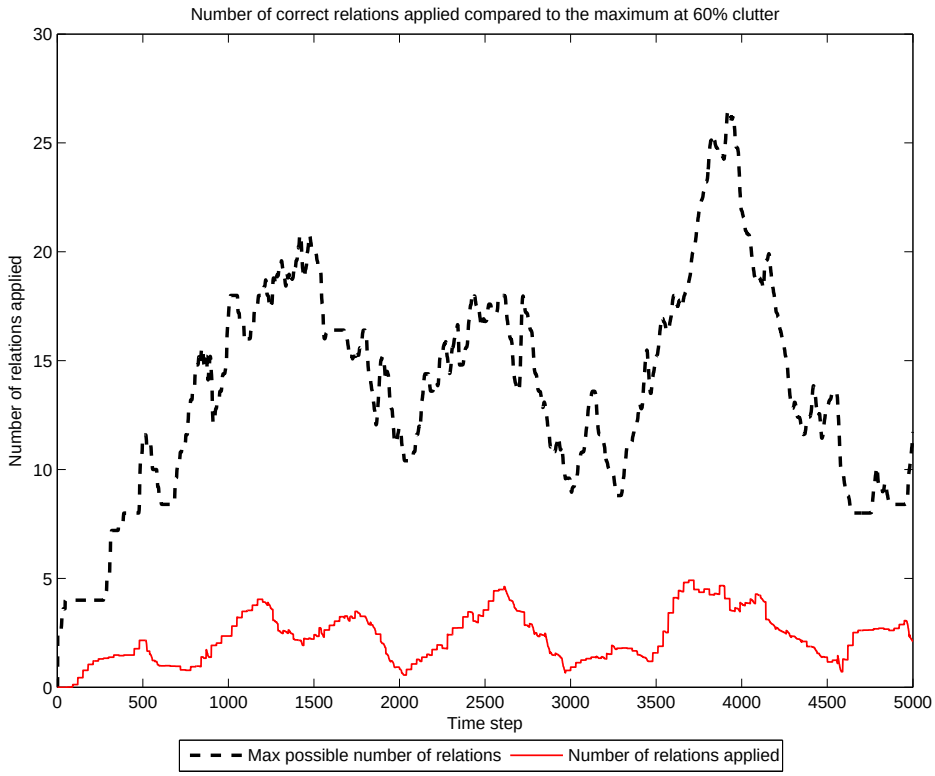

Figure 5.14: The number of relations applied over time for the 60% clutter case compared to the maximum number that could be applied at this clutter level if the relations were known. The relatively inaccurate prior map (1 m std. dev.) means that less than a third of potential relations are used.

uncertainty will be investigated in the future to remove the effect of clutter beacons as a possible cause of this effect. Ideally we would expect the decrease in the number of beacons with relations to correspond to the increasing clutter density, however the decrease is disproportionately greater. Figure 5.14 shows the number of correct relations applied for the 60% clutter case compared to the maximum number of correct relations that could be applied at this level of clutter. The number applied is less than a third of the number that could in theory be applied. This penalty comes from the difficulty in determining whether relations hold given the sparsity of the environment with the relatively inaccurate map (1 m std. dev. error), where few sets of beacons will be sufficiently correlated to meet this threshold. The number of relations applied could be increased by conditioning the prior for a relation holding on further sensor data using the framework in chapter 3.

Figure 5.15 shows the number of false positives; beacons with relations applied incorrectly (the beacons do not lie on the wall because the structure is not common). The number of such beacons is approximately ten times less than the number of correctly constrained beacons, for each clutter level (with the exception of the no clutter case). The clutter level does not appear to significantly affect the false positive rate, indicating that it does not affect the functioning of the geometric likelihood.

The rMSE and degree of inconsistency is still substantially lower for the vehicle operating in a 60% clutter environment compared to SLAM without a prior map, showing that even though this clutter

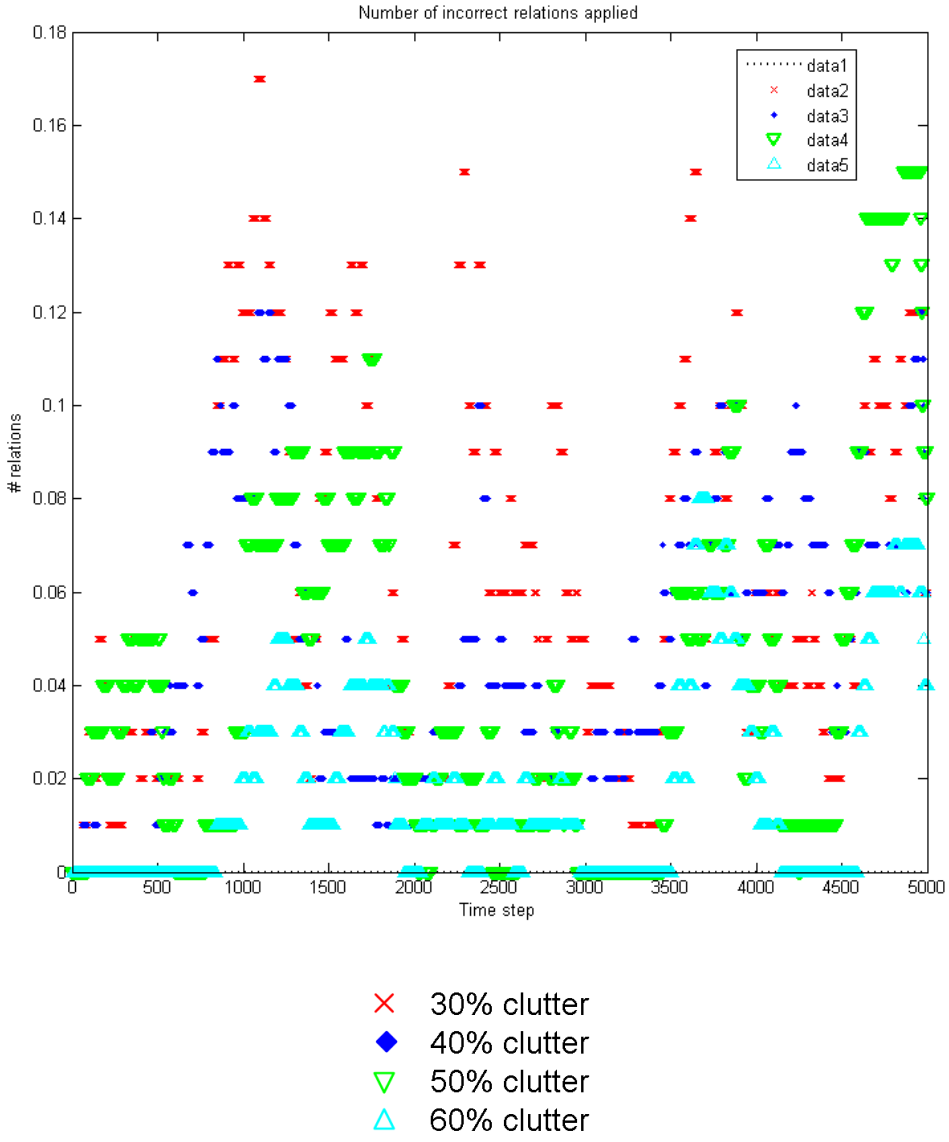

Figure 5.15: Number of cases where relations have been applied to beacons that do not lie on their line segment (the structure is not common, thus these are false positives) in the state over time. The number of relations is fractional because they have been averaged over 100 MC runs.

level results in significantly degraded performance over lower clutter levels (40% being the approximate threshold in this scenario), it does not affect the false positive rate, and the prior map is still useful. Even though the number of relations applied decreases nonlinearly with the clutter level, it appears that few relations need to be applied for each line segment for most of its information to propagate into the SLAM state.

The use of a more informative prior could significantly improve the number of relations applied at such clutter levels, potentially resulting in a localisation performance similar to that achieved at lower clutter levels.

5.3.4 Varying Prior Map Accuracy with a Constant Clutter Density

Figure 5.16 shows the effect of reducing the accuracy of the prior map on the vehicle pose while keeping the degree of clutter at 40%. As with the case of increasing clutter, the orientation error and uncertainty remains similar for all the levels of prior map error. Surprisingly the pose error for SLAM with a 50 cm prior map is slightly more accurate than with a 10 cm prior map. This is likely caused by the high number of relations being applied incorrectly in the 10 cm prior map case, as shown in Figure 5.19. The degraded performance suggests that for accurate prior maps, the geometric relation likelihood is more susceptible to EKF and linearisation errors.

The pose error for a 1 m prior map is generally similar to that of the 10 and 50 cm prior maps, however the error with the 1.5 m prior map is significantly worse. The number of relations applied correctly is similar to that of the 0.5 and 1 m maps, and the number of incorrect relations applied is lower, suggesting that this performance degradation is not due to a lack of relations or incorrect relations. Instead it appears that at this level of prior map accuracy and beyond the map information is weak enough that SLAM performance reduces and EKF and linearisation errors begin to dominate.

Figure 5.17 gives an indication of the severity of inconsistency in the vehicle pose over all the runs, and thus the likelihood that any run will remain within a given consistency criterion.

There is a significant improvement for all prior map uncertainty levels compared to regular SLAM. At 3σ regular SLAM only has 29%, whereas SLAM with the 10 cm prior map has 56%. At 6σ SLAM with all the prior map uncertainties exceeds 90%, whereas regular SLAM achieves just 63%. This shows that even at a 40% clutter level in our environment, using the prior map reduces the degree of inconsistency in the filter estimate.

The degree of inconsistency does not correspond to the prior map quality; the most inconsistent was for the most accurate (10 cm) prior map. The 50 cm and 1 m prior maps gave the least inconsistent pose estimate. This suggests that at accurate prior map levels consistency is significantly affected by nonlinearities and association errors.

Figure 5.18 shows that the number of beacons with correctly applied relations (the beacons actually lie on the wall) in the state at each time step does not vary as significantly with prior map accuracy as with clutter, except in the case of the relatively accurate 10 cm prior map, where significantly fewer correct relations are applied. This suggests that at the 40% clutter level the system is not “starved” of beacons with which to resolve relations.

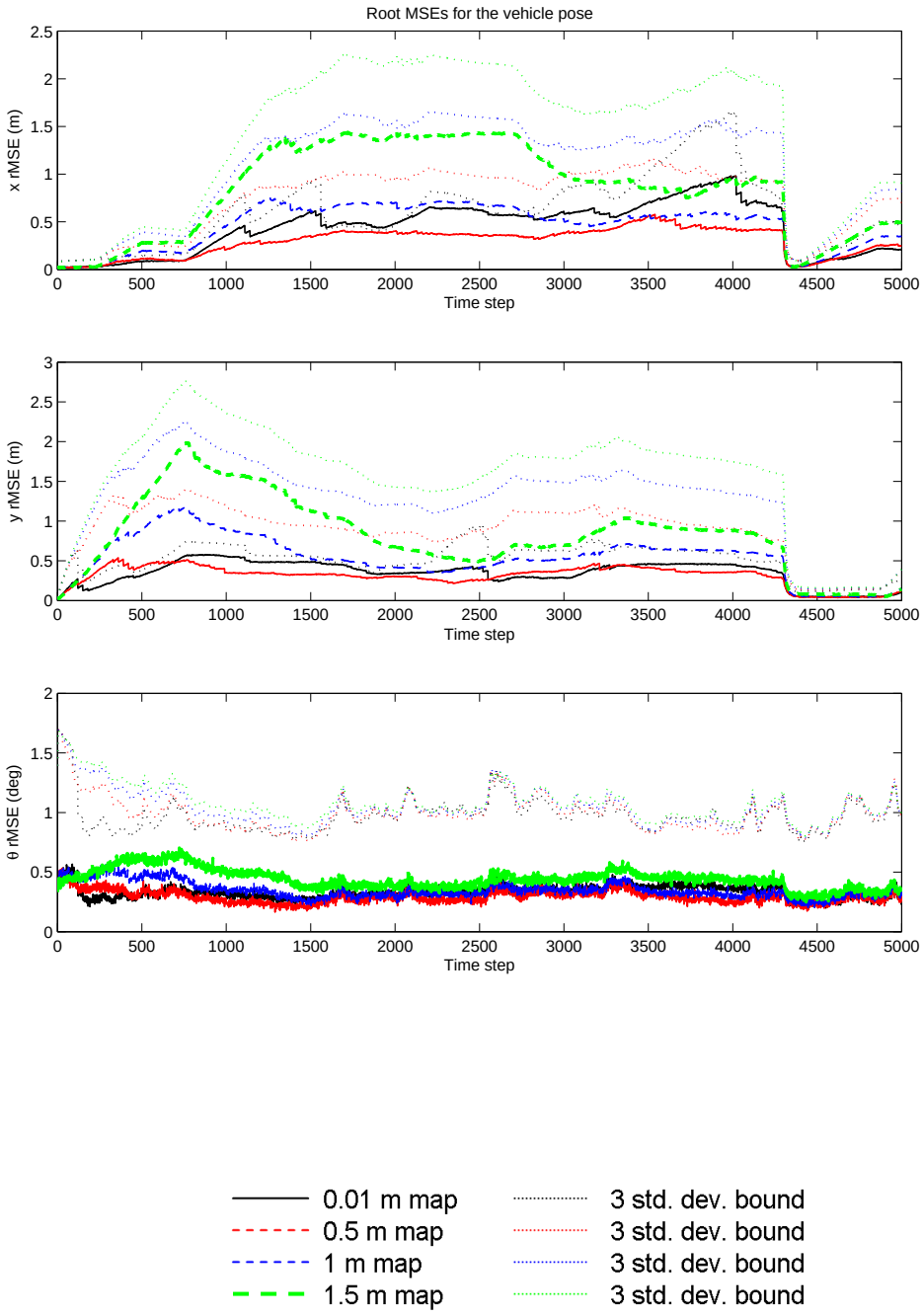

Figure 5.16: Root mean squared error (rMSE) of the vehicle poses with four prior map accuracy levels (1σ). From top to bottom x, y and θ. 3σ standard deviations are shown as dotted lines. Loop closure occurs at 4310 s. The average vehicle pose x, y and θ elements remain within their average 3σ bounds in all cases.

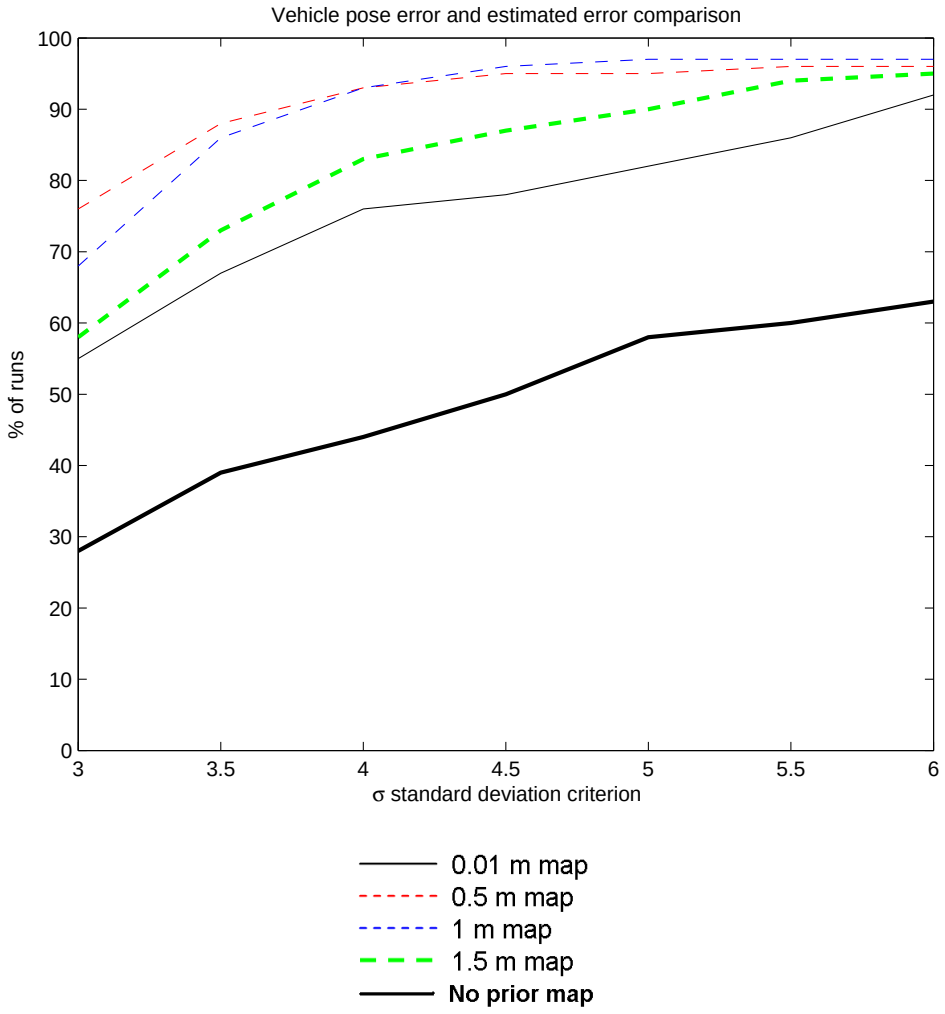

Figure 5.17: Comparison of the number of Monte Carlo runs where the vehicle pose remained within the given number of standard deviations (on the x-axis) for at least 95% of the run.

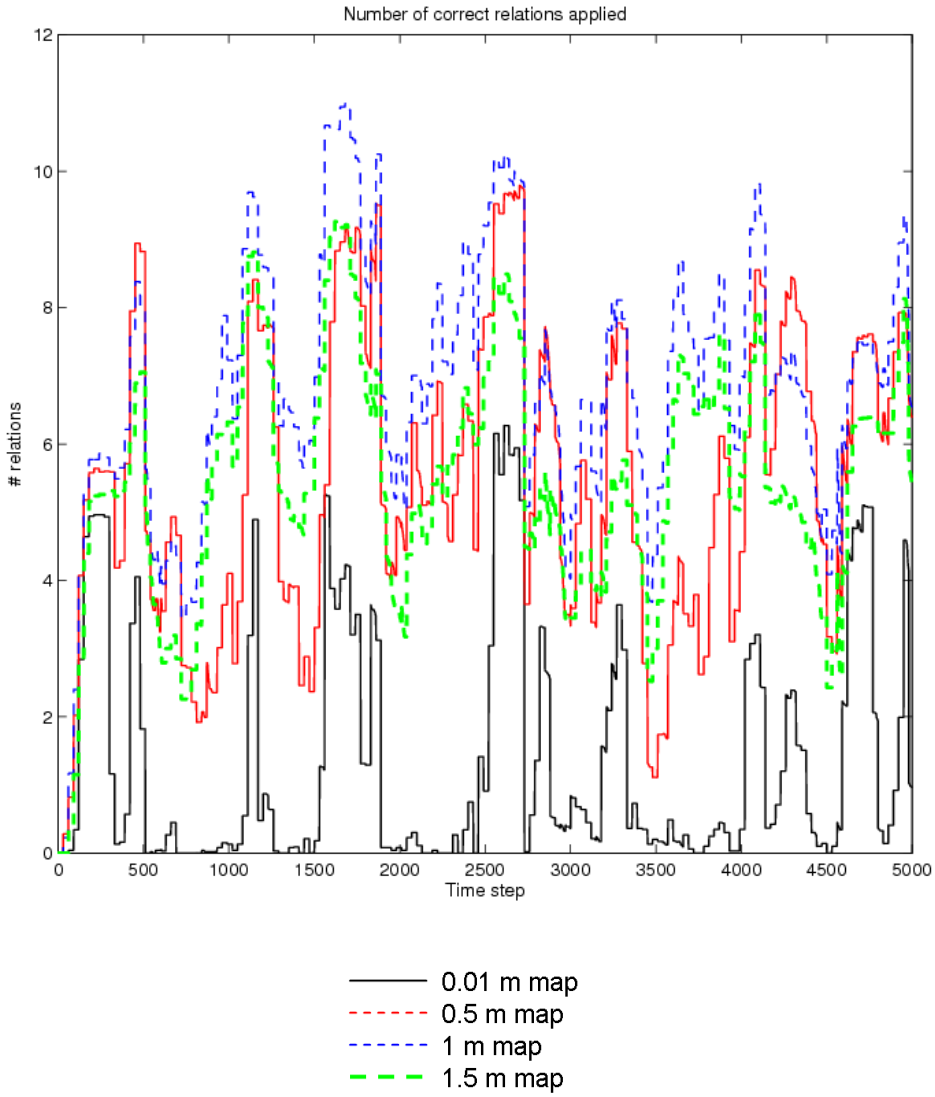

Figure 5.18: Number of beacons with relations correctly applied (the common structure is present) in the state over time.

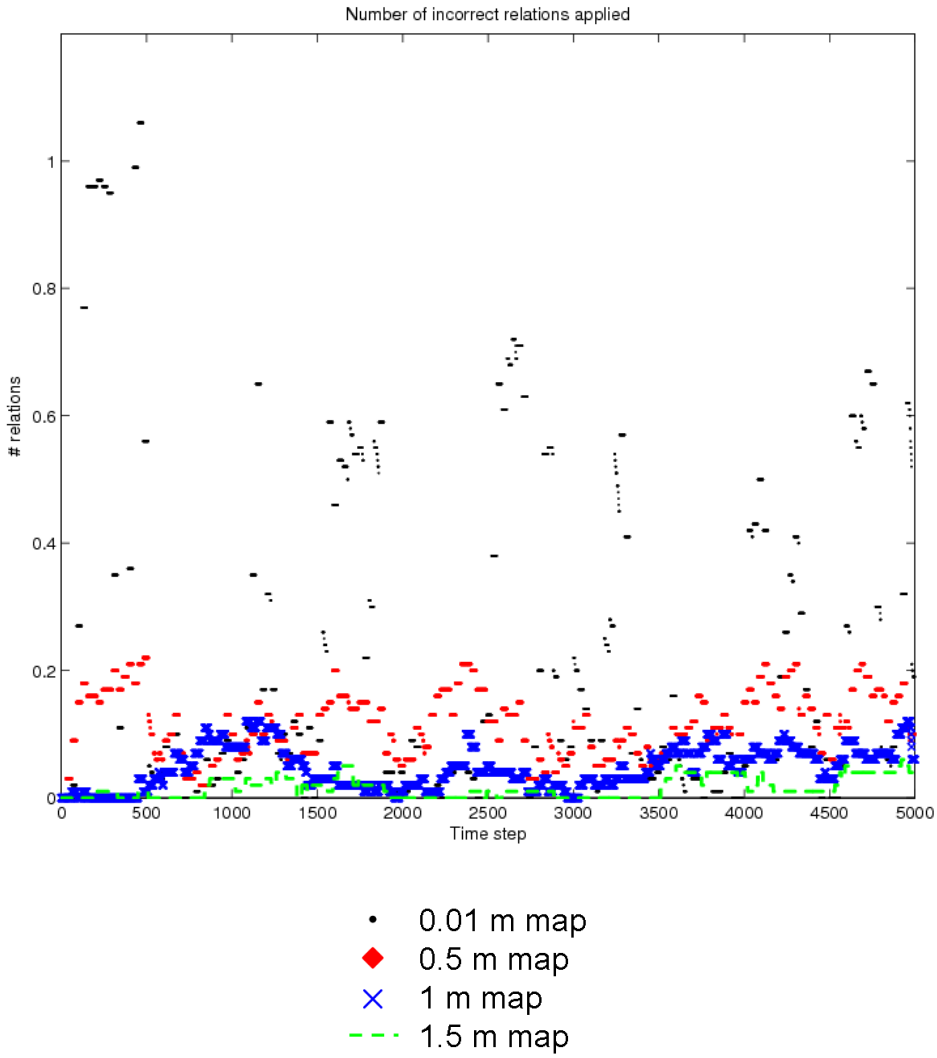

Figure 5.19: Number of false positives; beacons with relations incorrectly applied (the structure is not common) in the state over time.

Likewise the 10 cm prior map shows a greater number of incorrectly applied relations, as shown in Figure 5.19. For the less accurate maps the number of incorrect relations applied is far lower, and decreases with increasing prior map error. In general the normal distance of incorrectly constrained beacons to the wall is far less than 1σ for all but the most accurate prior map, where errors can exceed 3σ (30 cm). These results are likely to be because of the effect of EKF inconsistency on the likelihood of a relation holding. EKF map corruption in a local region affects the global position of beacons more than the correlations between them. Thus the low accuracy maps are more robust to these errors because the likelihood is more dependent on the joint configuration of multiple beacons than on their absolute position. Because of the structure of SLAM, the joint configuration is less affected by such errors.

Unlike the case of increasing clutter, where lower clutter means better localisation performance, a more accurate map does not necessarily result in better localisation performance. The geometric likelihood for very accurate prior maps appears to be more sensitive to EKF errors when resolving relations. As a result for very accurate maps the number of correct relations identified is lower than expected, and the number of incorrect relations (false positives) much higher, and the false positive rate decreases with increasing map error. Thus a more accurate prior map can result in worse localisation accuracy and consistency than one that is less accurate.

Also unlike the varying clutter case, the consistency of the pose estimate does not appear to significantly decrease as the prior map error level increases, for the levels of prior map error we considered. Thus SLAM with our least accurate prior map (1.5 m std. dev. error) significantly outperforms SLAM without a prior map.

5.4 Conclusion

The single state method is effective at determining whether relations hold using a geometry-based likelihood function, for a computational cost similar to single state EKF-SLAM. The method is effective even for a sparse map (with an average of 1 beacon per 5 m of prior map wall), where as many as 60% of the beacons do not lie on the walls (and thus relate with the prior map), or for a prior map with a 1σ error of 1.5 m where 40% of the beacons do not lie on the walls. This is within the error levels of many geometric map sources, for instance urban OS MasterMap data [54] and Mars Explorer imagery [81]. The effect of varying map accuracy and clutter levels is summarised in Table 5.2, showing how in all cases the absolute average error was better than that of Second Order EKF SLAM without a prior map.

The number of relations applied decreases nonlinearly with the clutter level, but appears invariant to the prior map accuracy level for our range of map accuracies. Beyond a certain clutter level, the localisation accuracy and consistency decreases significantly. The number of relations applied could be increased by conditioning the prior for a relation holding on further sensor data using the framework in chapter 3.

The geometric likelihood is sensitive to the prior map accuracy level, with an increase in the false positive rate (relations being incorrectly applied) with increasing prior map accuracy. This is likely due to the increased sensitivity of the likelihood function to EKF and linearisation errors as the prior map becomes more accurate. The false positive rate is not significantly affected by the clutter level.

Table 5.2: Summary tables showing the effect of clutter and prior map accuracy on the average absolute error of the vehicle pose over the entire run for the average of the Monte Carlo runs. In all cases the average rMSE was improved by exploiting the prior map compared to SLAM without a prior map.

Prior map 1σ uncertainty (m)	Average rMSE (m)	3σ consistency criterion (%)
0.01	0.68	76
0.5	0.54	68
1	0.83	58
1.5	1.35	55
No prior map	3.25	28

(a) Varying prior map uncertainty with a constant clutter level of 40%.

Prior map clutter level (%)	Average rMSE (m)	3σ consistency criterion (%)
No clutter	0.55	86
30	0.69	69
40	0.83	68
50	1.19	54
60	2.21	47
No prior map	3.25	28

(b) Varying clutter level with a constant prior map uncertainty of 1 m (1σ bound).

The retrospective association strategy in this chapter has a number of advantages. Only a single state needs to be maintained regardless of the number of relation hypotheses, thus saving computation and storage costs and implementation complexity. The approach is flexible, as one can choose when to exploit structure, whereas the MHSLAM approach (using the CSF) requires that the states of all hypotheses be enumerated and updated. This is an inefficient strategy given the exponential growth of MHSLAM and the CSF with the number of hypotheses and the fact that few hypotheses will ultimately survive. In addition, like MHSLAM the CSF requires the weights to be computed following every update, whereas by delaying the application of relations from structure we can evaluate the posterior distribution in a single time step at convenient intervals, then (with the exception of EKF errors and nonlinearity) optimally fuse the state and map once the correct hypothesis is known. We can also marginalise out problematic relations to increase our chances of finding a highly probable hypothesis. Thus unlike even the CSF, the computational cost can generally be kept manageable even when operating with sparse numbers of beacons, large uncertainty and high clutter levels; under these conditions it can take a long time to converge on the correct hypothesis.

Another disadvantage of an MHSLAM approach is that the representative (MAP) state is subject to fluctuate between different hypotheses over time as the posterior distribution is updated from observations. For some applications it may be more desirable to have an estimate that is known to be correct, even if conservative, in which case MHSLAM has little value over a single state method other than for mitigating EKF errors.

However, the single state retrospective approach also has a number of disadvantages. In the linear case, the posterior probability distribution over the structure relation hypotheses produced by the CSF and the single state method is the same, however in the more realistic nonlinear case linearisation and other errors will affect the single state method to a greater extent, because unlike MHSLAM it does not apply the relation information immediately. If significant map slip occurs, the map produced by the SLAM state will be unlikely, and thus unlikely to relate with the prior map. Because MHSLAM applies relations immediately, nonlinear effects can be mitigated or reduced immediately.

One could combine the CSF with the single state method presented in this chapter to obtain the best of both. For instance hypotheses may only be created once their number has been trimmed down to a manageable level. For the case of a few very informative relations it may be worth maintaining a few hypotheses concurrently and thus benefitting from the more accurate vehicle and map estimate.

There are a number of aspects of the method that may be improved in further work. The cause of the need for stabilising noise in the line segment observations needs to be investigated. The mechanics by which the Dual Representation is able to mitigate EKF errors also needs to be investigated. A method such as lazy data association [51] may be used to reverse incorrect associations and thus reduce the corruption that these cause.

In the next chapter we apply the method in this chapter to part of the Oxford City Centre data set, a real data set collected by a robot with a laser scanner in an urban environment.

Chapter 6

Experimental Results

6.1 SLAM with The Oxford Data Set

For evaluating the performance of our method on real data we use part of the Oxford City Centre data set [24]. This is an outdoor loop in a flat urban environment, shown in Figure 6.1 and 6.2, where GPS is unreliable and there is some dynamic motion. We assume that the environment is planar and thus do planar 2D SLAM with a 3 DOF (degree of freedom) vehicle.

This data set offers a number of challenges:

- There are many dynamic objects, including cars, people and bicycles. These produce non-static features which can corrupt the estimate if used in the SLAM process without explicitly accounting for their motion, due to the assumption of static features.

- The environment is outdoors and unstructured. Unlike indoor environments which tend to consist of planar rectilinear walls, the outdoor environment has many complex unpredictable shapes due to cars, foliage, pavements and other "clutter" in the environment. The buildings are generally at various angles with protruding or inset objects such as windows and doors. Thus the scans produced by the sensor are more complex and there are fewer stable features from scan to scan.

- The GPS is unreliable due to the "urban canyon" effect (to the extent that we do not use the GPS), and the physical configuration of the robot, being skid steer means there is high, correlated odometric noise.

- The LIDAR sensor we use produces a 2D planar scan, the plane shifting as the robot pitches and rolls (for instance on bumps). As the environment is unstructured and the sensor information is sparse, it is challenging to extract common features reliably across consecutive scans; indeed it is not even guaranteed that there will be any common features across consecutive scans for all the scans. In addition some of the scan points can contain spurious structure, for instance due to reflections from the ground when the robot pitches downwards.

- The high degree of clutter in the environment close to the building walls, for instance parked cars, means that it is challenging to determine whether a point feature in the SLAM state came from a wall structure or other structure near it when using a prior map of the building walls.

Figure 6.1: Aerial image showing the environment, with the approximate robot trajectory shown as a black line.

We run for 3 sides of the loop, for a total trajectory length of approximately 650 m. The last edge has few stable features and many dynamic objects (which we do not model) and thus feature extraction and SLAM fail to perform adequately.

6.1.1 SLAM Platform

The SLAM platform is a modified "ATRV-Jnr" wheeled robot called "Marge", shown in Figure 6.4. Although this platform has a nodding SICK scanner for 3D point cloud collection, an XSens inertial sensor and GPS, we only use its 2D (fixed) SICK scanner and the onboard odometry. The GPS data is highly inaccurate in the urban area of operation and is thus unused.

As the trajectory is relatively flat we performed 2D SLAM with a 3 DOF vehicle pose, instead of full 6 DOF SLAM. We did this partly to reduce the complexity associated with working with 3D data, such as aligning the scan slices into a common coordinate frame (compensating for the robot motion), and to make the real experiment consistent with the 2D simulations and prior map used throughout this thesis.

The robot motion model is

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k-1} + \triangle t \begin{bmatrix} (u_v(k) + v_v(k)) \cos(\varphi(k)) \\ (u_v(k) + v_v(k)) \sin(\varphi(k)) \\ u_\theta(k) + v_\theta(k) \end{bmatrix} + \mathbf{Q}_s, \tag{6.1}$$

where$\triangle t$ is the time elapsed between $k-1$ and k and $\varphi(k) = \theta(k-1) + (u_\theta(k) + v_\theta(k))\triangle t$, and $\mathbf{Q}_s$ is stabilising noise. $u_v(k)$ is the velocity from odometry with noise $v_v(k)$, and $u_\theta(k)$ is the orientation angular velocity from odometry with noise $v_\theta(k)$.

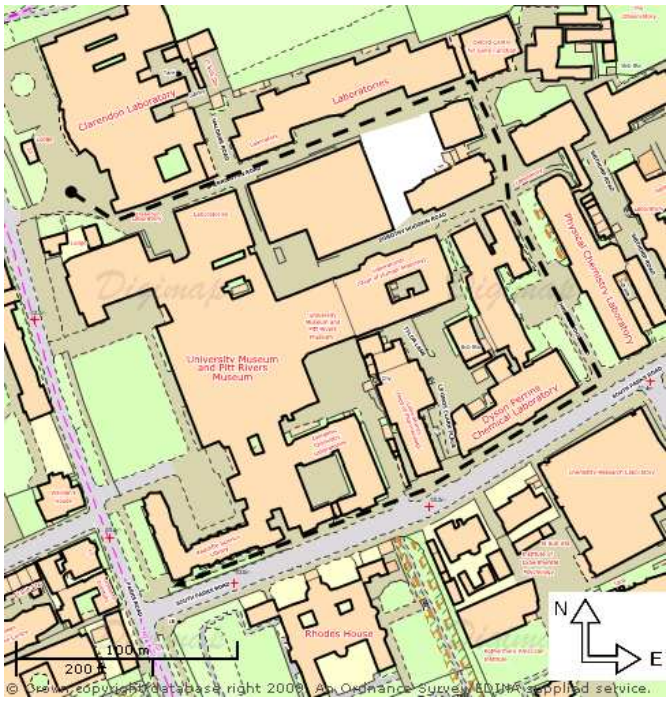

Figure 6.2: The OS MasterMap image from which the prior map was derived, with the approximate robot trajectory shown as a dashed line. We only selected the main long building wall line segments around the trajectory, to give a significantly sparser prior map. (Image source: Ordnance Survey/EDINA)

Because of biases introduced by wheel slip and inaccurate wheel radii, the velocity error is non-zero mean. We model the bias with an additive velocity correction factor $u_{\delta v}(k|k)$, which we estimate in the state, thus

$$v_v(k) \sim \aleph(u_{\delta v}(k), \sigma_v^2), \tag{6.2}$$

where $\aleph(\mu, \Sigma)$ is Gaussian distributed noise with mean μ and variance Σ. We assume that $u_{\delta v}$ is subject to zero-mean drift over time with standard deviation $\sigma_{\delta v}$,

$$u_{\delta v}(k) = u_{\delta v}(k-1) + \triangle t\, v_{\delta v}(k), \tag{6.3}$$

where $v_{\delta v}(k) \sim \aleph\left(0, \sigma_{\delta v}^2\right)$.

Table 6.1 shows the vehicle and sensor parameters used in the experiment.

6.1.2 Feature Extraction

Our SLAM implementation is based on using relatively sparse beacons extracted from 2D laser scans (from a SICK scanner). After extracting salient features we do not make any further use of the laser data; it is discarded.

We perform SLAM using three separate point feature beacon types based on 3 feature extraction methods; isolated points, point clusters and CPDA [52] points. Observations extracted using each

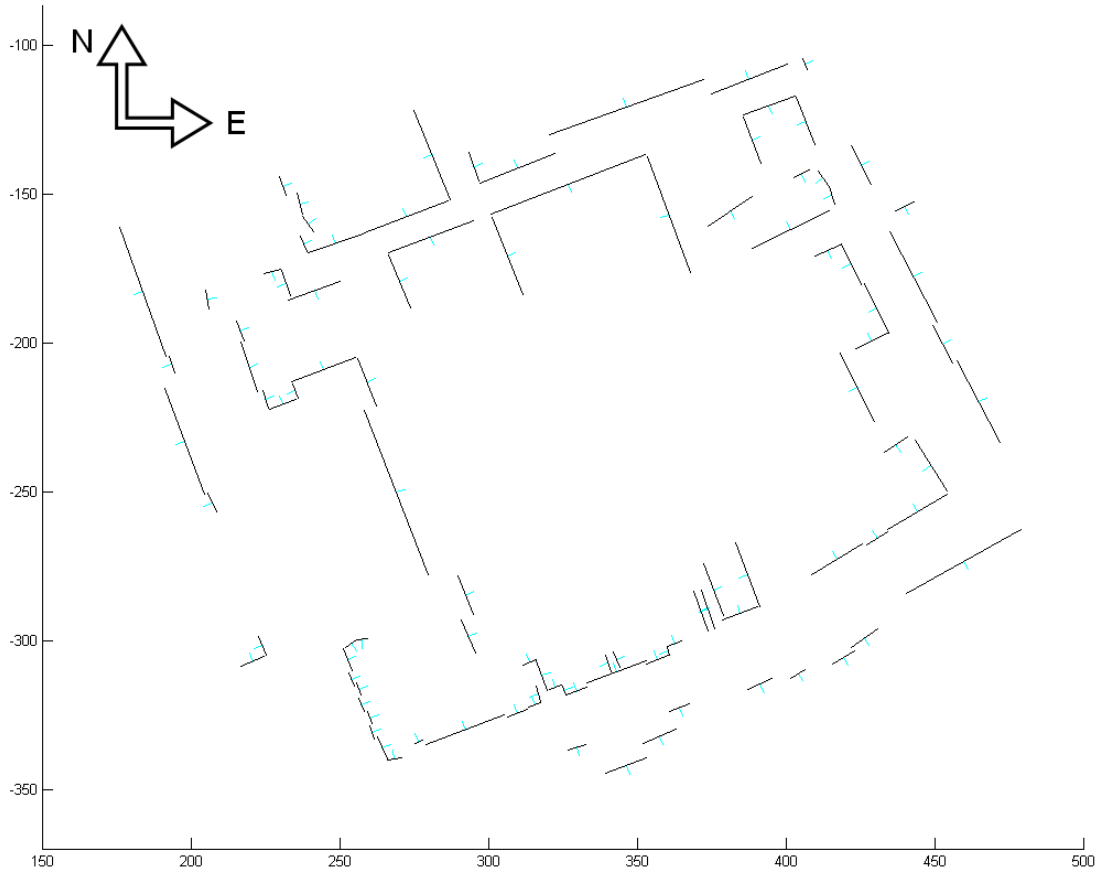

Figure 6.3: Prior map of line segments representing the main building walls, extracted by hand from the OS MasterMap image shown in Figure 6.2 (units in m). The short light lines indicate the direction into the building for each line segment.

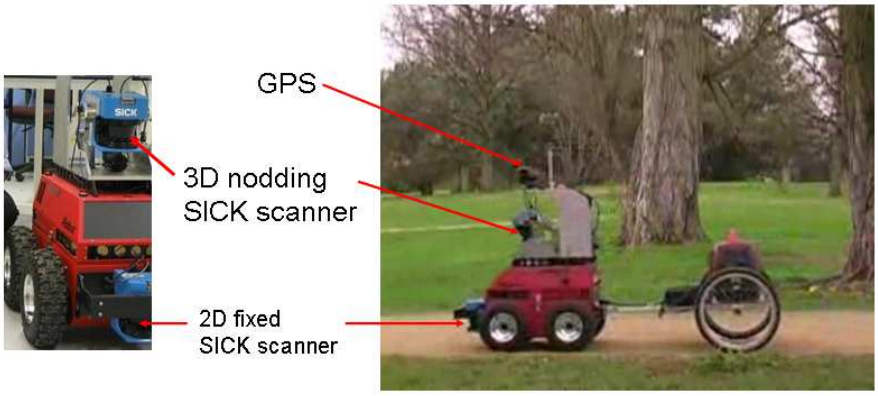

Figure 6.4: The "ATRV-Jnr" platform used to collect the data set. Note how close the 2D SICK scanner is mounted to the ground.

Table 6.1: Vehicle and sensor parameters for the Oxford data set experiment.

Parameter	Symbol	Value
Starting uncertainty	$\begin{bmatrix} P_x & 0 & 0 \\ 0 & P_y & 0 \\ 0 & 0 & P_\theta \end{bmatrix}_{0\|0}$	$\begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.09 \end{bmatrix}$ $(\mathrm{m}^2, \mathrm{deg}^2)$
Speed noise	σ_v	0.1 m/s
Speed bias parameter drift rate	$\sigma_{\delta v}$	0.2 m/s
Orientation angular velocity noise	σ_θ	3°
Stabilising noise	Q_s	$3.5 \times 10^{-5} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Maximum range	r_{max}	approx. 30 m
Maximum sweep	ϕ_{max}	$\pm 90^\circ$
Range error	σ_r	0.03 m
Bearing error	σ_ϕ	0.4°
Prior map element-wise covariance	P_m	1 m^2

method can only gate with a beacon type corresponding to that method. Figure 6.5 shows a typical scan with examples of the three types of features extracted from it.

Feature extraction on this data set is challenging for several reasons. Being an outdoor urban environment, there are insufficient obvious salient point features to extract (such as points corresponding to corners), and abundant clutter. Figure 6.6 shows an example of four typical consecutive scans, showing the degree of clutter and appearance of apparent spurious structure in two of the scans; these are probably returns from the ground. The SICK scanner is mounted very close to the ground, and will often pick up returns from the ground when the robot dips, as also shown in Figure 6.7. In addition, there are many dynamic objects such as cyclists, cars and pedestrians. Figure 6.8 shows an example of features being picked up as a cyclist cycles past the robot, while Figure 6.9 shows features extracted from moving cars on South Parks road.

6.1.2.1 Cluster Features

We group the laser points into clusters, based on a cut-off of 0.2 m in range difference between bearing-wise successive points. We require these clusters to meet several criteria to be considered. The range separation between successive clusters must be greater than 1 m, and each cluster must consist of between 1 and 15 points. In addition, the range of all points in a cluster must be over 2 m. Figure 6.10 shows these thresholds diagrammatically for two clusters. Clusters that do not meet these criteria are removed.

The mean of the range and bearing of compliant clusters are considered salient cluster points. Salient cluster points that are closer than 1 m to each other are removed, and the rest are returned as observations.

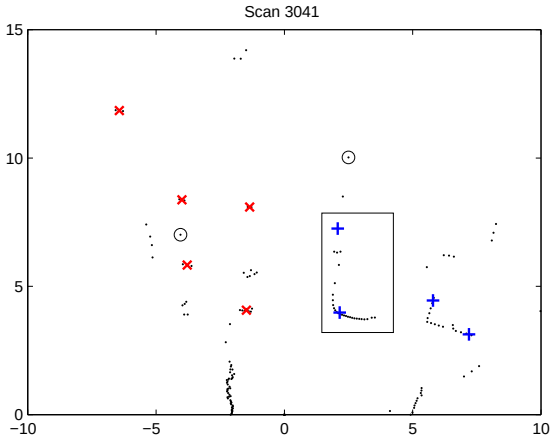

Figure 6.5: Example laser scan with three types of extracted features; isolated points (circles), point clusters ($\times$'s) and CPDA points ($+$'s). The dots represent the raw laser range points, the scanner being at (0,0), forwards being in the y-axis direction. The shape in the rectangle corresponds to a car driving towards the robot.

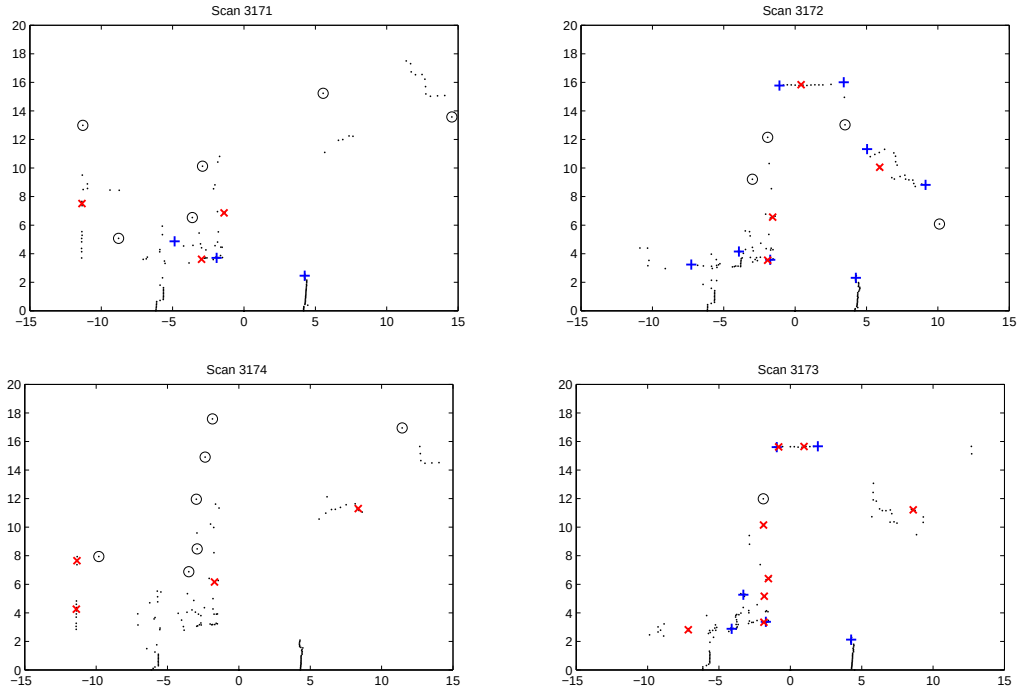

Figure 6.6: Top going clockwise: four successive scans showing the degree of clutter and spurious features typical of the environment. In two of the scans a vertical line that appears to be a wall appears, however this is not present in prior or subsequent scans and is probably caused by ground returns as the robot dips. Such spurious features contribute to making reliable feature extraction on this data set non-trivial.

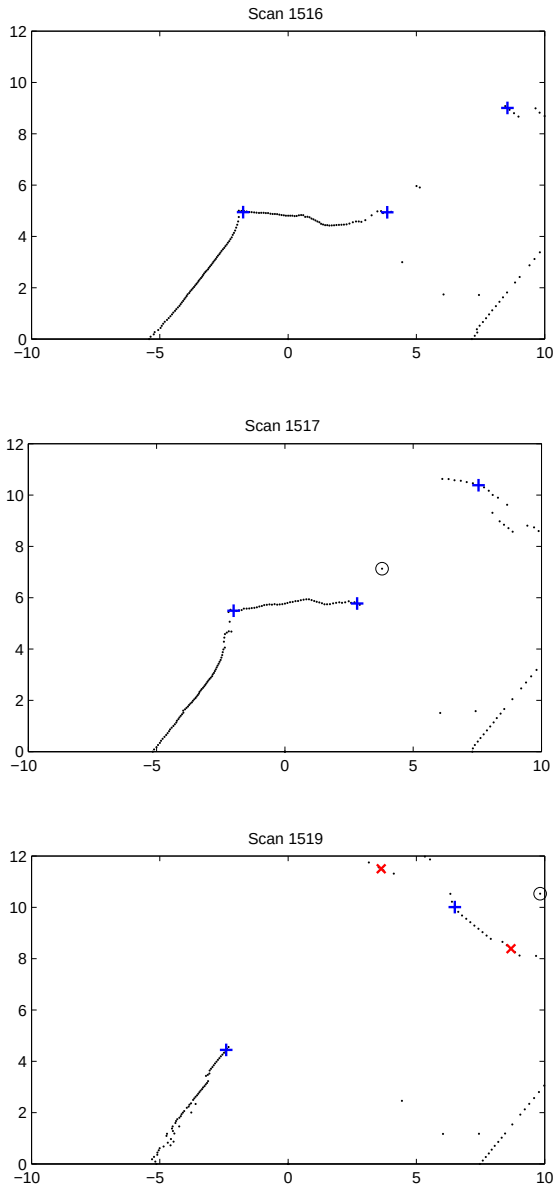

Figure 6.7: Top and middle: spurious points on two consecutive frames being picked up from the ground as the robot dips. Bottom: a scan where the ground returns are not present.

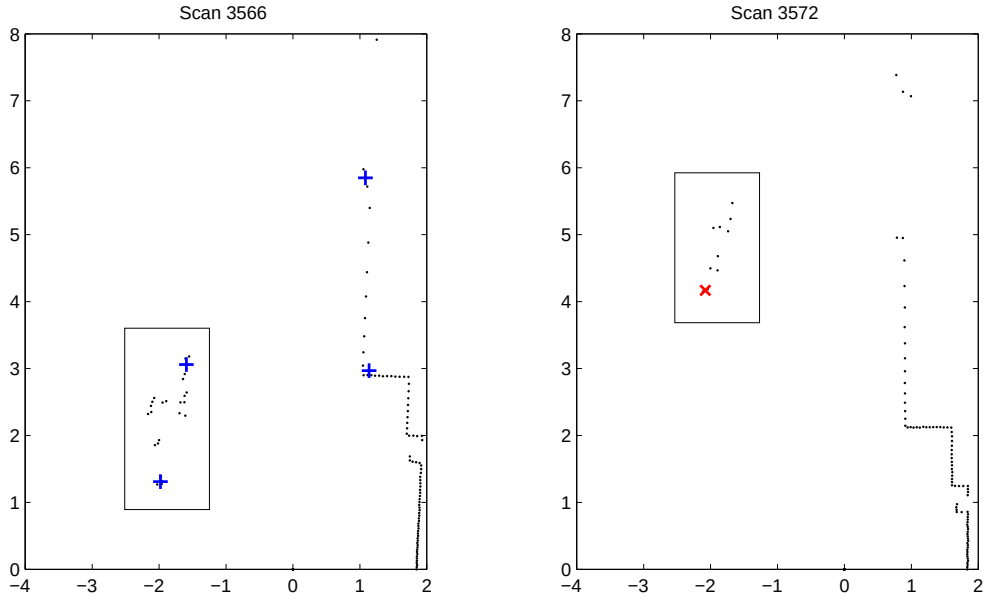

Figure 6.8: Features picked up as a cyclist cycles past the robot (shown in a rectangle). We do not explicitly detect moving objects, instead relying on such features being filtered out as clutter.

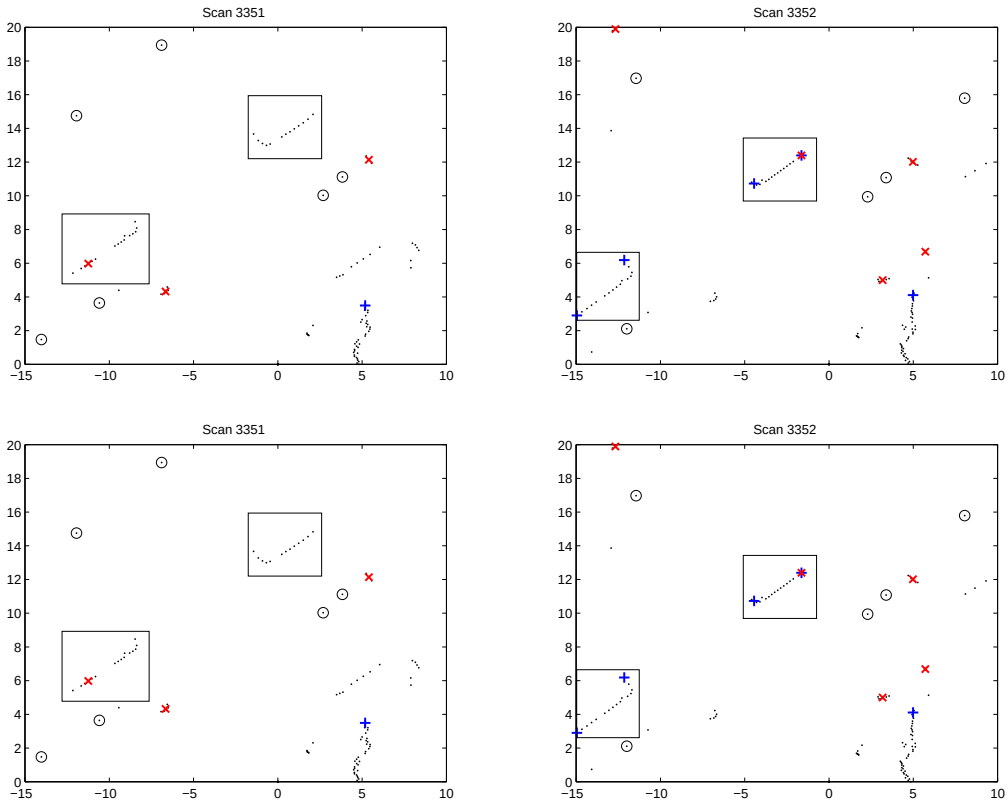

Figure 6.9: Features picked up on moving cars (shown in rectangles) as the robot approaches South Parks road.

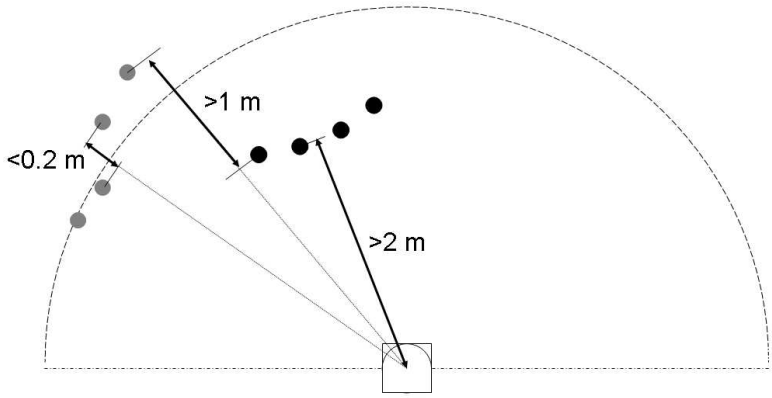

Figure 6.10: Extraction of two separate clusters showing the inter and intra-cluster separation thresholds required for these to be valid clusters. Each point within the cluster must be within a range of 0.2 m of its neighbour, and points separating beacons must be more than 1 m apart. In addition, points in clusters must be more than 2 m away from the sensor.

6.1.2.2 Isolated Point Features

We seek to find isolated laser scan points that are potentially salient, for instance because they correspond to vertical poles. To find these we consider the Euclidean distance between the laser points in a scan. Points that are closer than 1 m to any other point, or within a 1 m exclusion zone are removed, as shown in Figure 6.11. The exclusion zone removes points that may only appear to be isolated because they lie in the periphery of the scanner.

We must also exclude the effect of walls, which produce a characteristic non-salient pattern whereby points far from the scanner may isolated, but are not caused by to the presence of a salient feature in the environment, but merely the sparsity of points. To do so we filter triplets of bearing-wise successive scan points looking for colinearity, and remove those that appear to be colinear.

Compliant points are returned as salient point observations. Figure 6.12 shows an example of stable isolated point features picked up from bollards along South Park road, next to a wall showing the characteristic spacing of successive scan points that indicates these returns are from a wall, and thus unstable. Most of these points have been rejected as isolated point features due to their strong colinearity. The bollards have not been rejected as the laser points are not successive bearing-wise (there are other points in between that are not visible as they are further away).

6.1.2.3 Features from the Chord-to-Point Distance Accumulation (CPDA) Detector

We use the corner detector from [5], which is based on the chord-to-point distance accumulation (CPDA) [52] of an edge boundary. Such edges could correspond to the corners or ends of buildings and other objects in the environment. The CPDA of an edge boundary, which is a robust measure of its discrete curvature, is based on summing the perpendicular distance between a chord (line between two points on a curve) and the points in between, as shown in Figure 6.13. Thus for a point p_k on an edge and a chord of length L, the CPDA h_L^k is given by

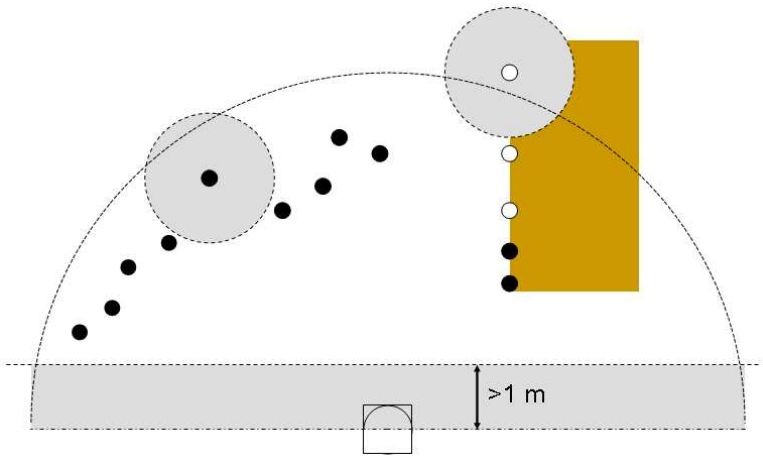

Figure 6.11: Extraction of isolated points. The thick dots represent laser scan points. The grey strip at the bottom indicates the 1 m exclusion zone from which we do not consider isolated points. The two shaded circles indicate potential isolated points; there are no beacons within 1 m of them. However whereas the dark dot on the left is a valid isolated point feature, the point on the right is discarded because it forms a colinear triplet with the bearing-wise neighbouring scan points (shown as white dots), indicating that it probably lies on a smooth wall and thus does not correspond to a stable feature in the environment.

$$h_L^k = \sum_{i=k-L}^{k} D_{ik}, \tag{6.4}$$

where D_{ik} is the Euclidean normal distance between the point p_i on the edge boundary and the chord (line) between the points p_{k-L} and p_k.

We extract salient points using CPDA as follows. First we form an occupancy grid for each scan at a resolution of 0.2 m per cell to form a binary image. Continuous edges in this image are then identified, smoothed with a small-width Gaussian kernel to remove quantisation noise, and the CPDA is computed for them for three chord lengths. Candidate corners are identified based on the local maxima of the computed CPDA values, and weak corners are removed by thresholding. The result of this process is that a number of cells corresponding to (salient) corners are returned, one cell per corner, as shown in Figure 6.14.

A salient point is derived from each cell by computing the mean of all the scan points that lie within 0.4 m of the centre point of the salient cell, as shown in Figure 6.15. We reject salient points within 1 m of the periphery of the scanner as the scan cut-off is often misinterpreted as corner features.

6.1.3 Observations

Most features produced from the feature extractors do not correspond to salient features. Thus clutter management is essential to filter out points that are not useful. Thus we remove beacons from the state if they are seen less than twice in the first 2 seconds after initialisation, and 3 times in the first 5 seconds.

In addition, to reduce data association ambiguities we aim to keep beacons at least 1 m apart, by not initialising new beacons if they would be closer than 1 m to an existing beacon in the map.

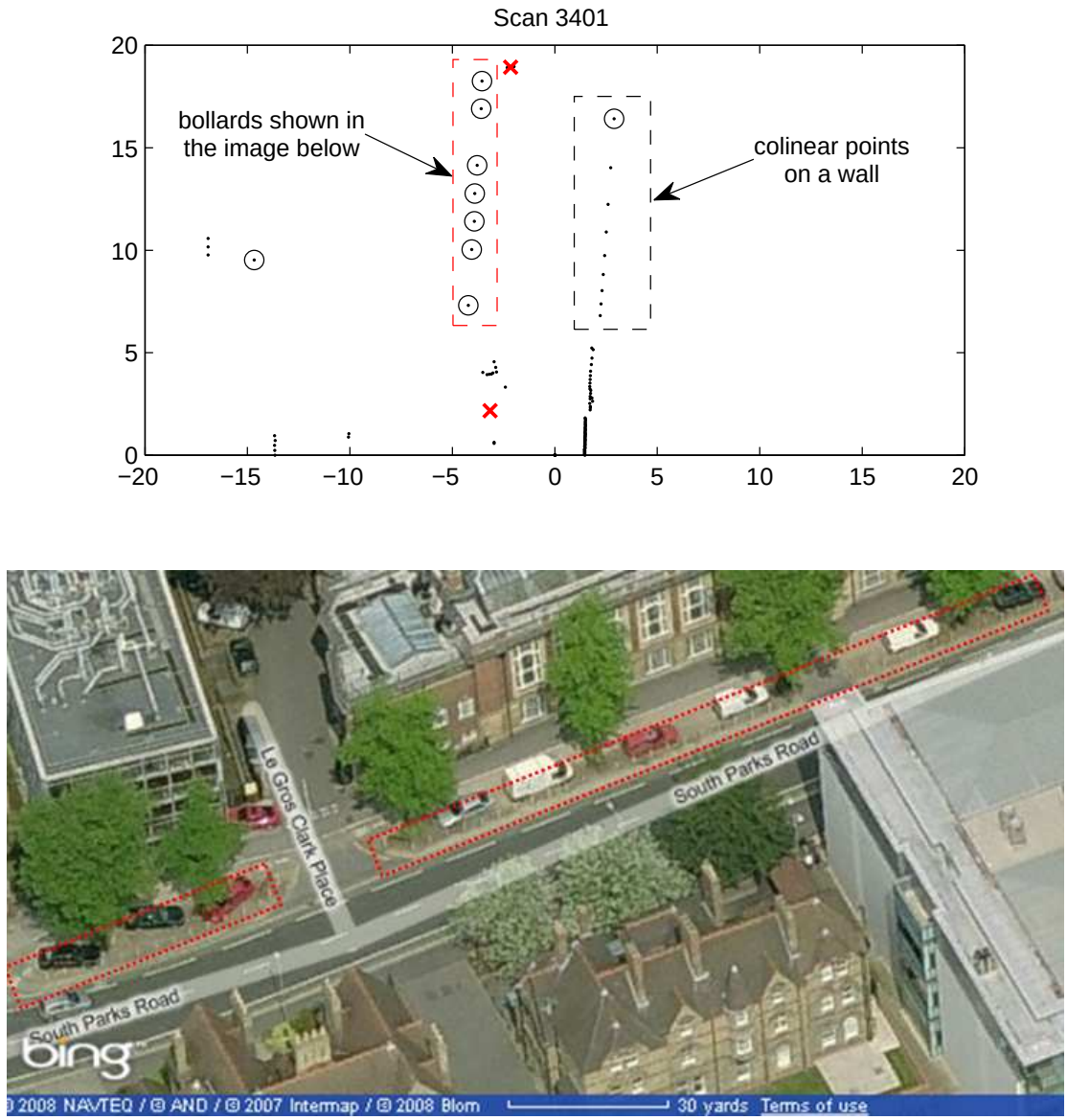

Figure 6.12: Top: laser scan image with extracted features corresponding to bollards and a generally smooth wall, both shown in the bottom aerial image. The bollards generate stable features, whereas those from the wall are unstable and return a fixed range; we seek to detect such features based on their colinearity and filter them out. (Image source: www.multimap.com)

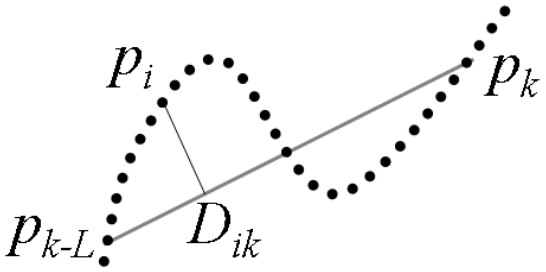

Figure 6.13: For the discrete set of points lying on an edge boundary, the CPDA of the point p_k with respect to a chord length L is the sum of the normal Euclidean distances between the points in between p_{k-L} and p_k and the chord between those points (shown as a thick grey line); this distance is shown for the point p_i by the thin black line.

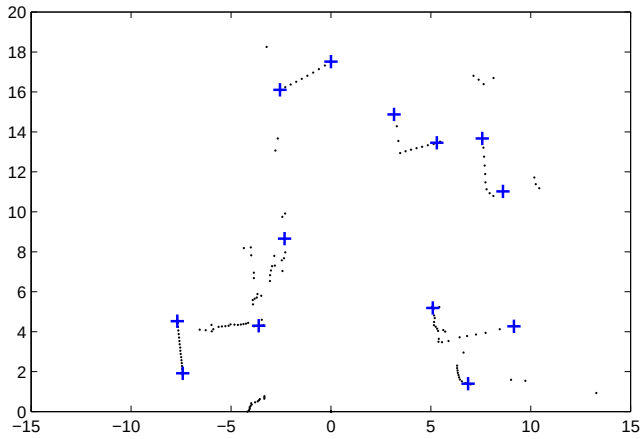

Figure 6.14: Features corresponding to corners and end points, extracted using the CPDA detector. Note that the sparsity of the scan means that corners are not always detected.

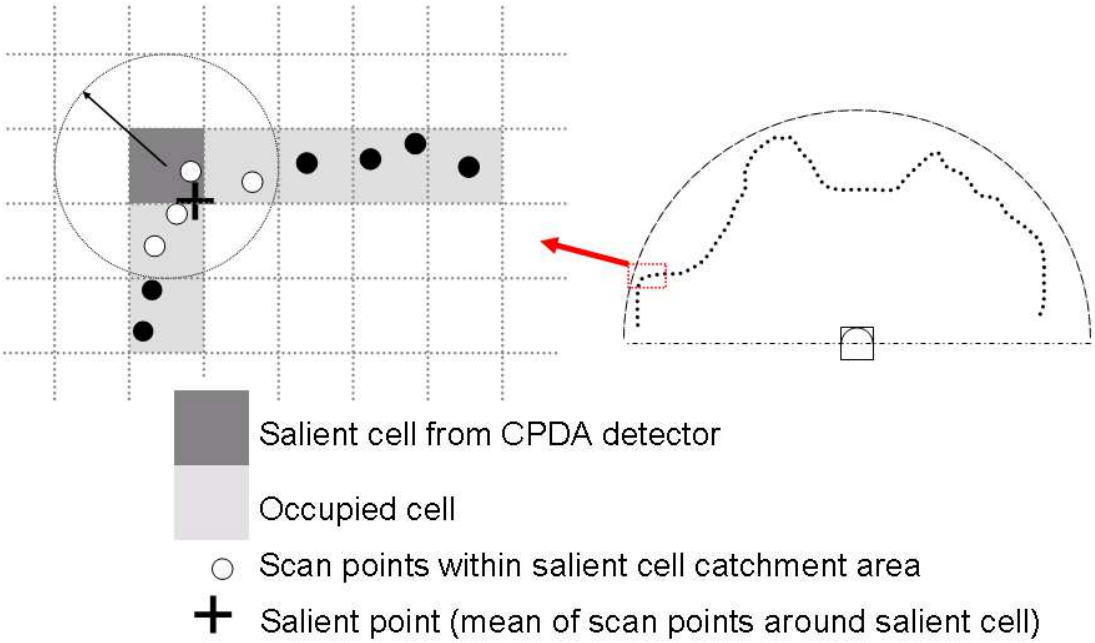

Figure 6.15: Diagram showing how a salient point (the cross) is extracted from the scan using the CPDA detector. An occupancy grid is formed at a resolution of 0.2 m per cell, and is processed as a binary image by the CPDA detector, which identifies salient cells. The salient point is then the mean of the set of scan points found within a certain radius (indicated by the arrow) of the centre of the salient cell (these points are shown as white circles).

Because we do not perform advanced map management techniques such as submapping, to keep the computational complexity manageable we remove all beacons not seen within 100 seconds from the state.

The range-bearing beacon observation model is the same as that described in chapter 2. We use joint compatibility branch and bound (JCBB) [88] to perform data association. JCBB searches for the set of beacon-observation pairings $\{\mathbf{i}, \mathbf{j}\}$ with the highest cardinality that also meet the criterion that the Mahalanobis distance of the error between the pairings lies within a given χ^2 bound (we use a 95% bound); thus they are considered *jointly compatible*. If there are ambiguous associations (multiple pairings with the same cardinality), we choose the set with the lowest Mahalanobis distance. By forming an interpretation tree, JCBB is able to bound the search for pairings $\{\mathbf{i}, \mathbf{j}\}$, significantly reducing the size of the search space.

6.2 Using the Prior Map in SLAM

6.2.1 Prior map processing

The prior map consists of line segments representing the major walls of buildings in the environment, and was extracted by manually overlaying line segments onto a raster image of an OS MasterMap map of the area, shown in Figure 6.2. For every line segment the side that corresponds to the building interior is manually identified. The prior map is shown in Figure 6.3. The OS MasterMap (urban) data has an average 1σ error of 0.6 m according to [54] and just over 0.3 m according to [94], however because we extracted the line segments from a lower resolution raster image[1] we assume the 1σ error in our line segments to be 1 m and independent.

6.2.2 Implementation

We use the single state method in chapter 5 to determine which features in the SLAM have a shared structure with features in the prior map, and apply it robustly using the Dual Representation.

We use the Gaussian likelihood function from (4.17) to evaluate (5.1) every 40 time steps, accepting the MAP structure indicator hypothesis if it exceeds a posterior probability acceptance threshold $p_{accept} = 0.5$, this giving the best results. Thus the MAP hypothesis with a posterior probability greater than 0.5 is accepted as correct; otherwise none of the hypotheses are accepted and their structure indicators are considered ambiguous until the next evaluation.

For computational efficiency we only consider beacons that gate with a line segment as potentially being part of a wall structure. To reduce the effect of clutter, beacons can only gate with a line segment once they have been observed at least 4 times. A beacon gates with a line segment feature if any of its 3σ ellipse lies within the 3σ bound normal to the line segment.

To reduce the influence of clutter from dynamic objects such as cars we tried performing SLAM while removing all beacons that did not gate with a prior map line segment (i.e. those that are not close to a wall). This could reduce data association errors under the assumption that structures close to walls tend to be static and produce less clutter than areas such as grass and roads. However we did not see an

[1] Unfortunately MasterMap GML data was not available in time for the experiment.

improvement in the results, and thus we did not use this method. This was because there is significant clutter from parked cars and other objects near to the building walls, in addition to the walls themselves, while some features not lying near the walls are stable and improve the data association process. Thus the complexity of the environment and the characteristics of the scanner mean that it was not clear a priori which features would be stable based on their proximity to the wall.

We use the likelihood function from (4.17) to evaluate (5.1), with a prior computed according to (5.6) using a heuristic-based model of building wall appearance as a classifier.

Point beacons have a prior probability $p(d(\mathbf{x}_{p\{i\}}, \mathbf{s}_{w\{s\}})) = 0.4$ of being part of the wall structure instance s, as they are themselves unlikely to have a common structure instance with a prior map line segment. As walls are persistent, we require all beacons to also be persistent rather than clutter (we consider beacons to be non-clutter if they have been seen at least 4 times). This reduces the computational and false positive effect of considering spurious beacons.

From the point beacons we detect agglomerated point features. These are wall-like features that are represented by a group of four or more regular point beacons, in a similar manner to the agglomerated features in [76]. These agglomerated features meet certain criteria that give them wall-like characteristics, and thus have a high conditional prior $p(d(\mathbf{x}_{p\{i\}}, \mathbf{s}_{w\{s\}})|\mathbf{z}_p) = 0.96$ (the prior referring to the fact that this is the probability prior to conditioning on the prior map observations $\mathbf{z}_m$) of being part of a wall structure, and corresponds to the prior in (5.1). The criteria for beacons in agglomerated point features are the following:

- The walls in the environment that we are interested in are all planar (straight lines in 2D space). Thus all beacon estimates $\hat{\mathbf{x}}_{p\{i\}}$ within an agglomerated point feature must be compatible with colinearity. We evaluate this as follows. Consider a hypothetical line $\mathbf{x}_{l\{n\}}$ with high uncertainty going through the first and last beacon estimate. The normal distance of each beacon to this line is given by (3.11), and would be zero if the true beacon positions are colinear. If this is the case, as in data association gating the Mahalanobis distance of the estimate would be expected to lie within a certain value 95% of the time, corresponding to the 95% confidence bound of the χ^2_c distribution with c degrees of freedom, where $c = \dim(\nu)$, $\nu = \mathrm{f}(\hat{\mathbf{x}}_{p\{i\}})$ and the covariance of ν is $\mathbf{S}$,

$$\nu^T \mathbf{S}^{-1} \nu < \chi^2_c|_{95\%}. \tag{6.5}$$

  Note how although we use this gating to evaluate whether there are four or more features compatible with being colinear, we do not use the value of the Mahalanobis distance in the prior value, as the Gaussian likelihood function from (4.17) implicitly considers the colinearity of the points (along with the uncertainty of the beacons and the prior map line segment).

- Walls are solid and planar in the region around a feature that lies on them. Thus all beacons that are part of an agglomerated point feature must be furthest into the building in their local region; that is if a beacon lies on a wall, while local beacons can lie in front of it, they cannot lie behind it. We assume the local region to be within 7 m of the beacon estimate as shown in Figure 6.16.

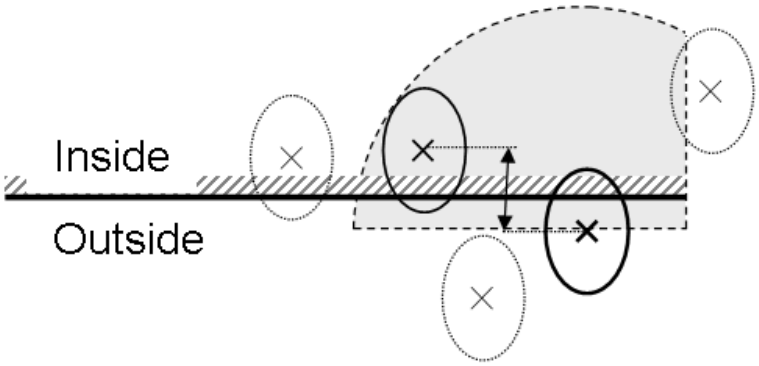

Figure 6.16: For a beacon whose estimate (shown in bold) indicates that it may be part of a wall structure (whose prior map estimate is shown by the horizontal line), we consider beacons that lie within a 7 m radius behind it and along the wall (shown by the highlighted region). If any such beacons lie more than 0.5 m further into the building than the beacon (as indicated by the arrow), we consider the beacon to not be part of an agglomerated wall structrue.

A beacon is assumed to lie behind another if its estimate is greater than 0.5 m behind that beacon in the direction normal to the wall. Because the building-side of each line segment in the prior map is marked, we know in which direction to look for such beacons.

6.3 Results

Figure 6.17 shows the run with pure odometry. By the end of the run the vehicle is over 50 m away from its actual position. To gauge the accuracy of the map we project the raw scan points from the laser scanner into the global frame using the vehicle pose estimate, showing the line segment prior map for comparison. By inspection the prior map agrees with aerial imagery, suggesting that the 1 m 1σ error we have assigned it is representative of the true error.

We tried to use the inertial sensor to improve the quality of the odometry, however we found it to hinder rather than help performance, even when used in a limited capacity to help in corners. Because the vehicle is skid steer, we would expect the odometry to be accurate in straight lines, but unreliable when cornering. Here the vehicle odometry tends to overestimate the angle by which the vehicle has turned, as shown in Figure 6.17. We tried switching to the INS (inertial navigation system) angular velocity when the steer angle exceeded a threshold, however the INS did not help in this regard (after having compensated for the INS bias at the start of the run), and thus we did not use the INS data at all.

Figures 6.18 and 6.19 give an overview of the entire trajectory. Note the presence of clutter that the laser has picked up, such as parked cars, grass and moving cars on South Park road. Comparison of the maps shows the improvement when performing SLAM with a prior map.

Figure 6.20 compares the maps at the top and bottom right corners of the trajectory. At the first (top right) corner, SLAM without a prior map error has accumulated approximately 5 m error, growing to around 8 m by the second (bottom right) corner, whereas SLAM with a prior map has accumulated around 1 m error at both corners. The vehicle pose uncertainty is correspondingly lower for SLAM with

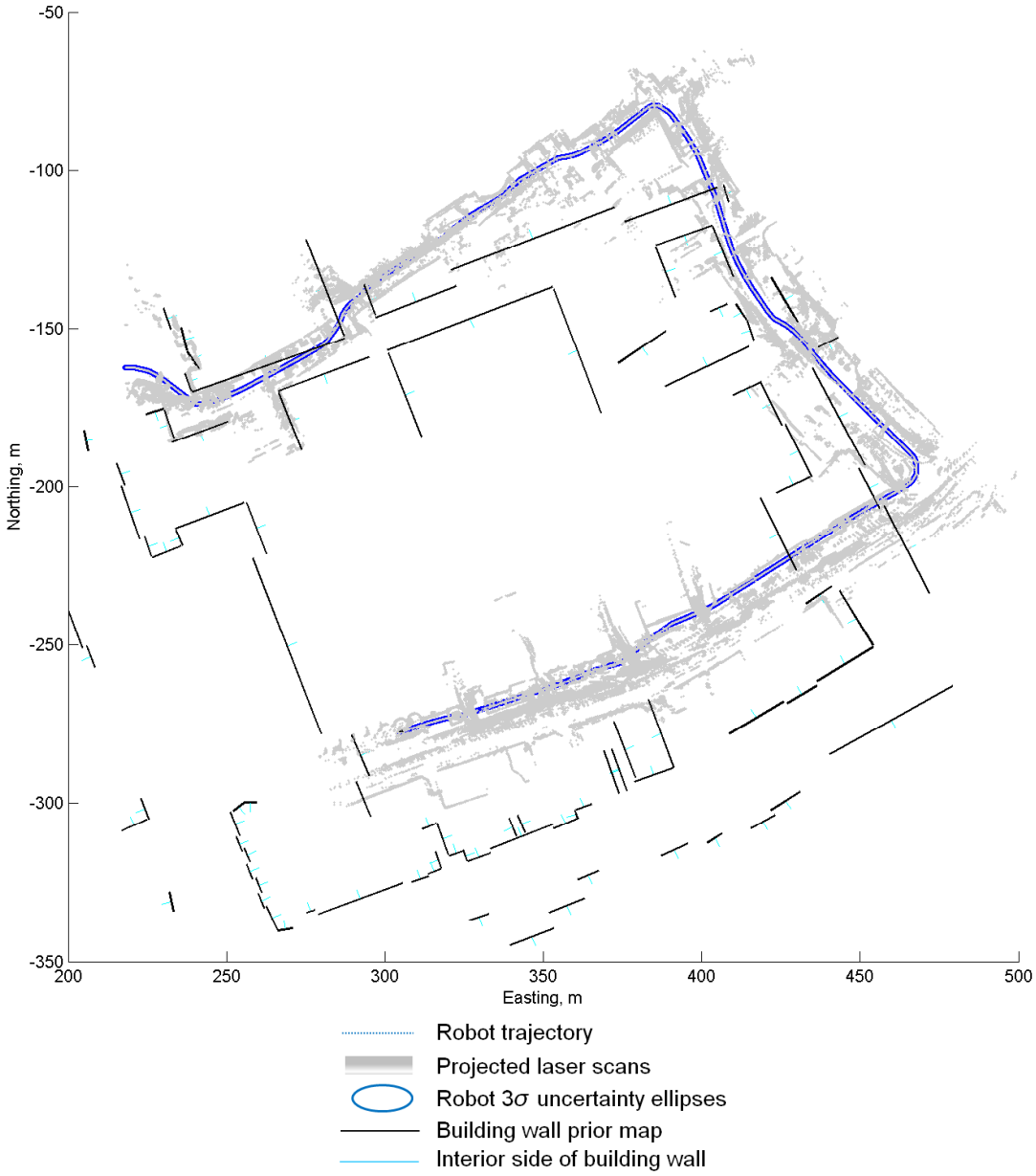

Figure 6.17: Trajectory estimate using pure odometry. Note how the path drifts away from the roads, appearing to go through the buildings in the prior map.

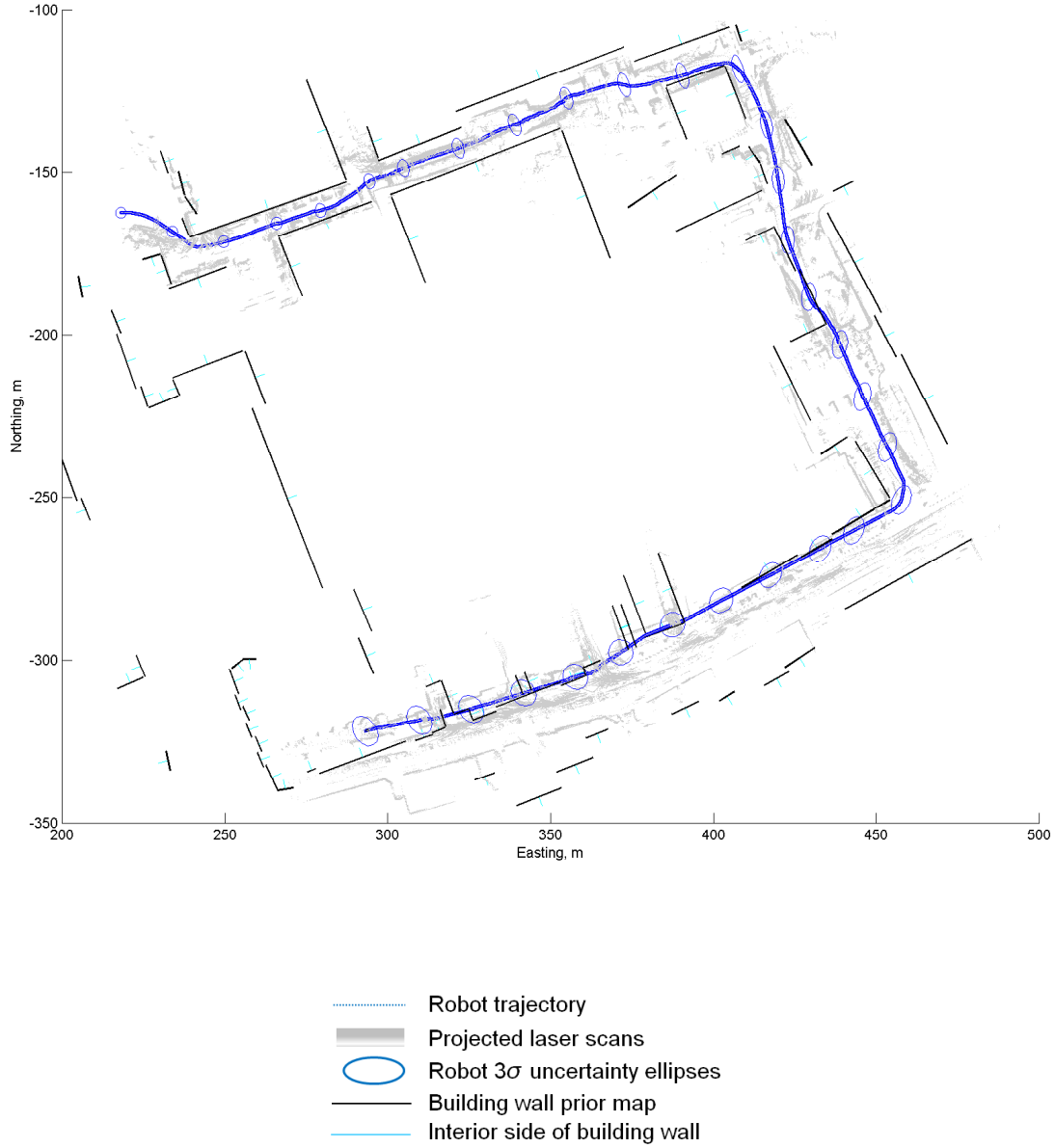

Figure 6.18: Map produced by SLAM without a prior map.

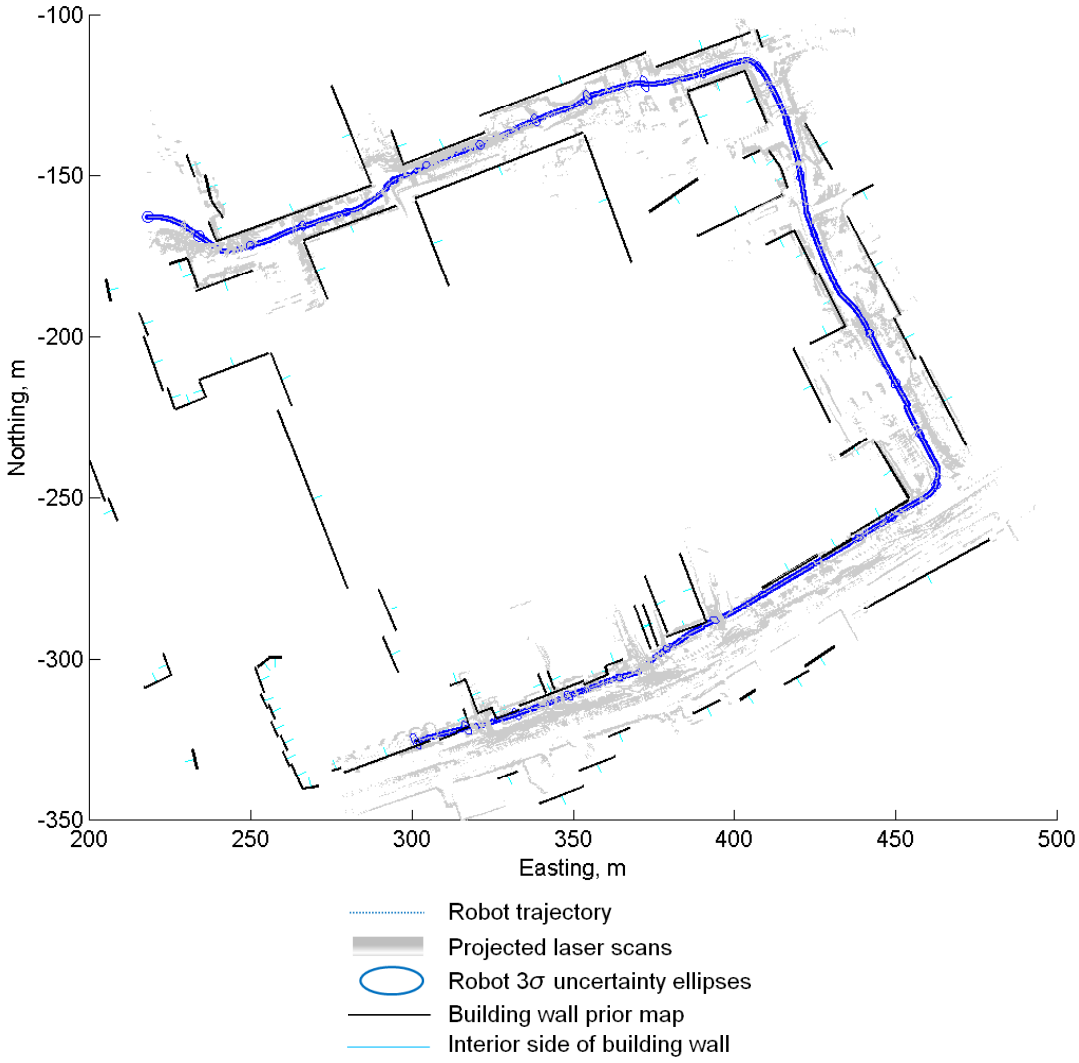

Figure 6.19: Map produced by SLAM with a prior map.

a prior map, as shown by the smaller uncertainty ellipses. Thus the use of the prior a prior map has considerably reduced the error and uncertainty in the map and vehicle pose.

Figure 6.21 compares the entrance to the building in the middle of the right edge of the trajectory, between the two corners. Without the prior map the entrance is poorly aligned, with an error of around 5 m in the x and y directions. However with the prior map the entrance is well aligned, and the robot pose uncertainty is smaller.

Figure 6.22 shows how both SLAM with and without a prior map have built up error by the end of the run, however the error for SLAM with a prior map is lower, as is the uncertainty in the robot position. In both cases the robot position estimate falls short of the position indicated by the prior map; this occurs gradually from the bottom right corner to the end. This is likely due to odometric bias, possibly due to the pavement environment on the bottom edge of the trajectory, compared to the prior tarmac surface. Figure 6.23 shows a moving average plot of the velocity bias estimate over time, with the time steps corresponding to the bottom section of road highlighted; the time taken to traverse this section is approximately 5 minutes. Note how the mean value decreases here, then sharply increases again. Few observations and dynamic objects on this part of the trajectory could cause the velocity bias estimate to fall below its actual value here. No associations with prior map line segments lying normal to the direction of travel were made on this part of the trajectory, and thus the unmodelled error in the direction of travel remains uncorrected and the estimate is clearly optimistic in this direction.

6.4 Conclusions

Even with our battery of feature extraction algorithms, SLAM on this dataset is challenging because of the lack of stable, repeatable points in the environment, and data association errors caused by spurious observations from clutter and dynamic objects, in addition to the EKF errors described in chapter 2.

However the experiment shows that even using sparse point features, a prior map can be deployed, with the effect of improving the accuracy of the SLAM estimate. This benefits are attainable even though the prior map is very inaccurate (1 m error) compared to the sensor on the robot (3 cm range error), and the fact that information from only some of the line segment features in the prior map was used.

By detecting which beacons in the SLAM map are likely to correspond to wall structures we are able to selectively tailor priors on there being a common underlying structure between features in the SLAM and prior map. Thus we do not have to rely on the geometric likelihood as the sole means of inferring whether underlying structure holds, this being of value as the uncertainty in the prior map and vehicle estimate are too high to successfully infer the structure without the informative prior.

There are a number of improvements that could be made. Firstly, dynamic objects could be identified and filtered out, from a combination of their movement and classification of the point cloud to exclude potentially dynamic object types, such as people. This could allow for the entire loop with loop closure to be performed. Secondly, GML data should be used directly, rather than manual extraction of lines from a raster image. Thirdly, the prior could be conditioned on more accurate models of building walls informed by surveys of buildings, rather than the crude models we have used. This would make use of the dense 3D laser scans, rather than just the sparse extracted point beacons we use. Poten-

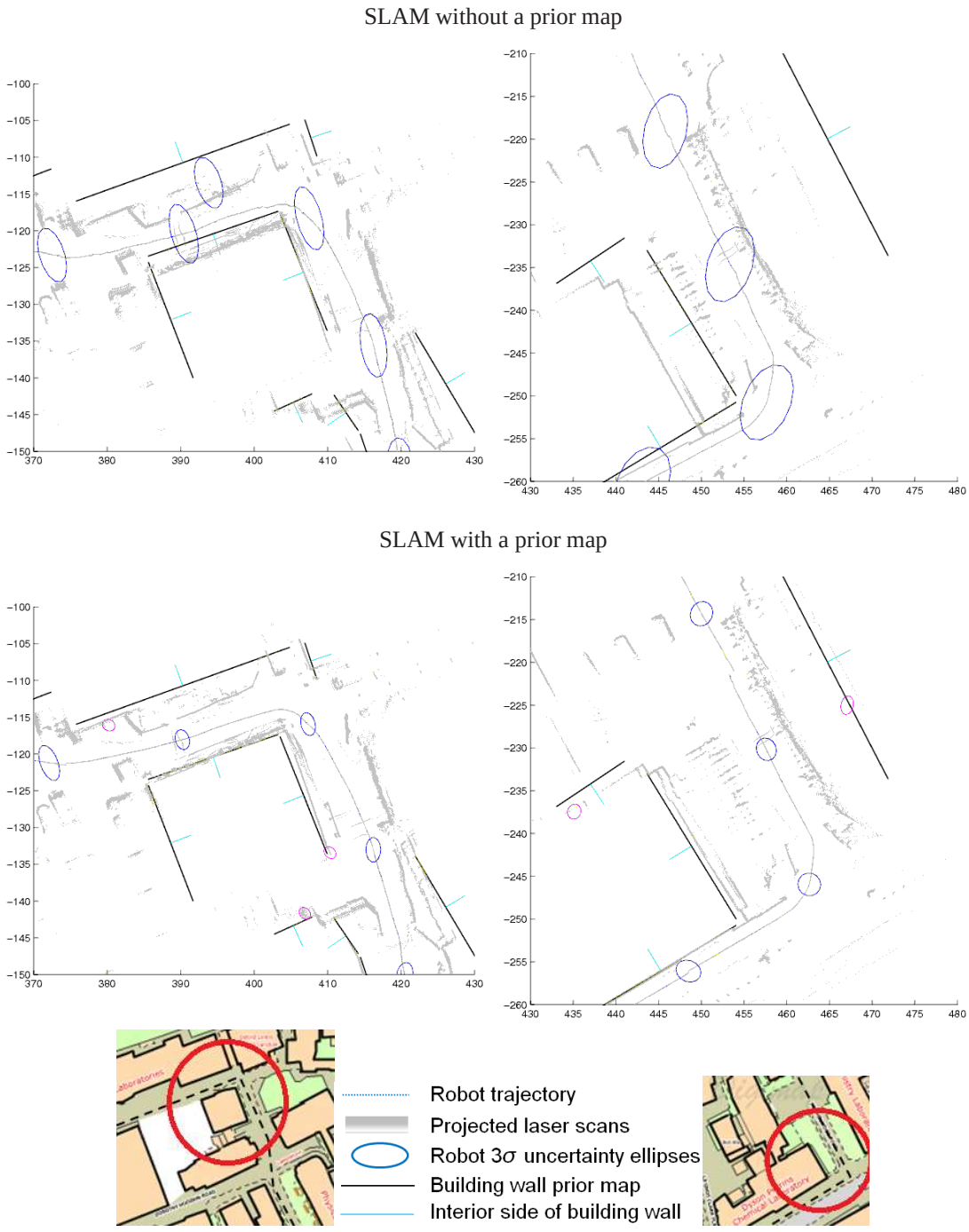

Figure 6.20: Top: SLAM without a prior map. Bottom: SLAM with a prior map for the top right and bottom right corners of the trajectory. The blue ellipses represent the 3σ robot position uncertainty at equally spaced intervals on the trajectory - the smaller ellipses for SLAM with a prior map indicate that the uncertainty estimate is smaller.

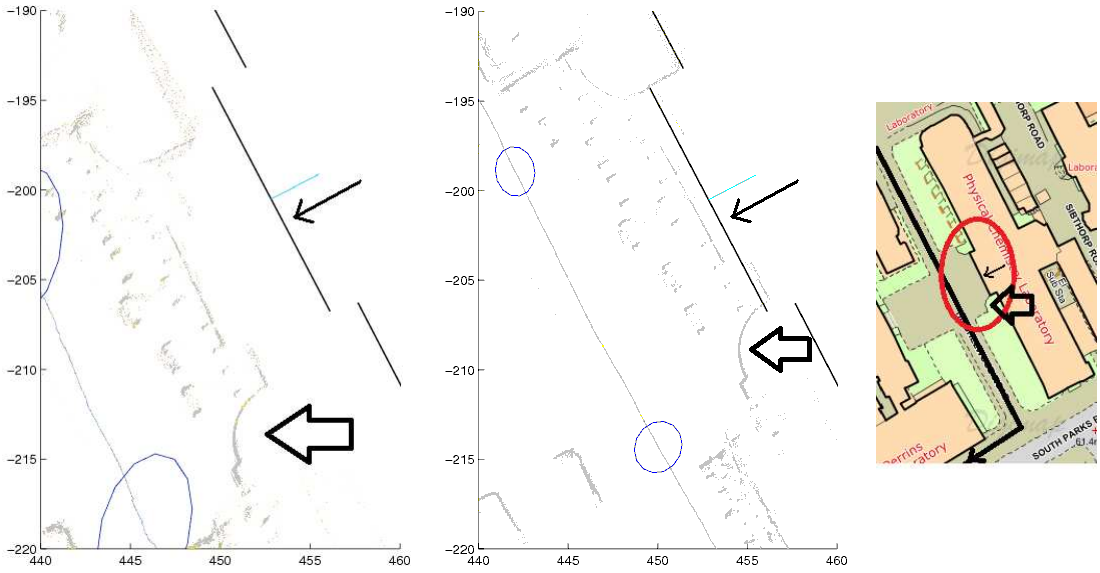

Figure 6.21: Alignment of the point cloud with the prior map for the entrance to the building in the middle of the right edge of the trajectory, for SLAM without (left) and with (right) a prior map. The blue ellipses represent the 3σ robot position uncertainty at two points on the trajectory, the grey points the projected scan points. The arrow shows the position of the grass verge indicated in the OS MasterMap to facilitate comparison between the two maps.

tially information from other sensor modalities could be used to complement this, for instance by using observations from a camera mounted on the robot.

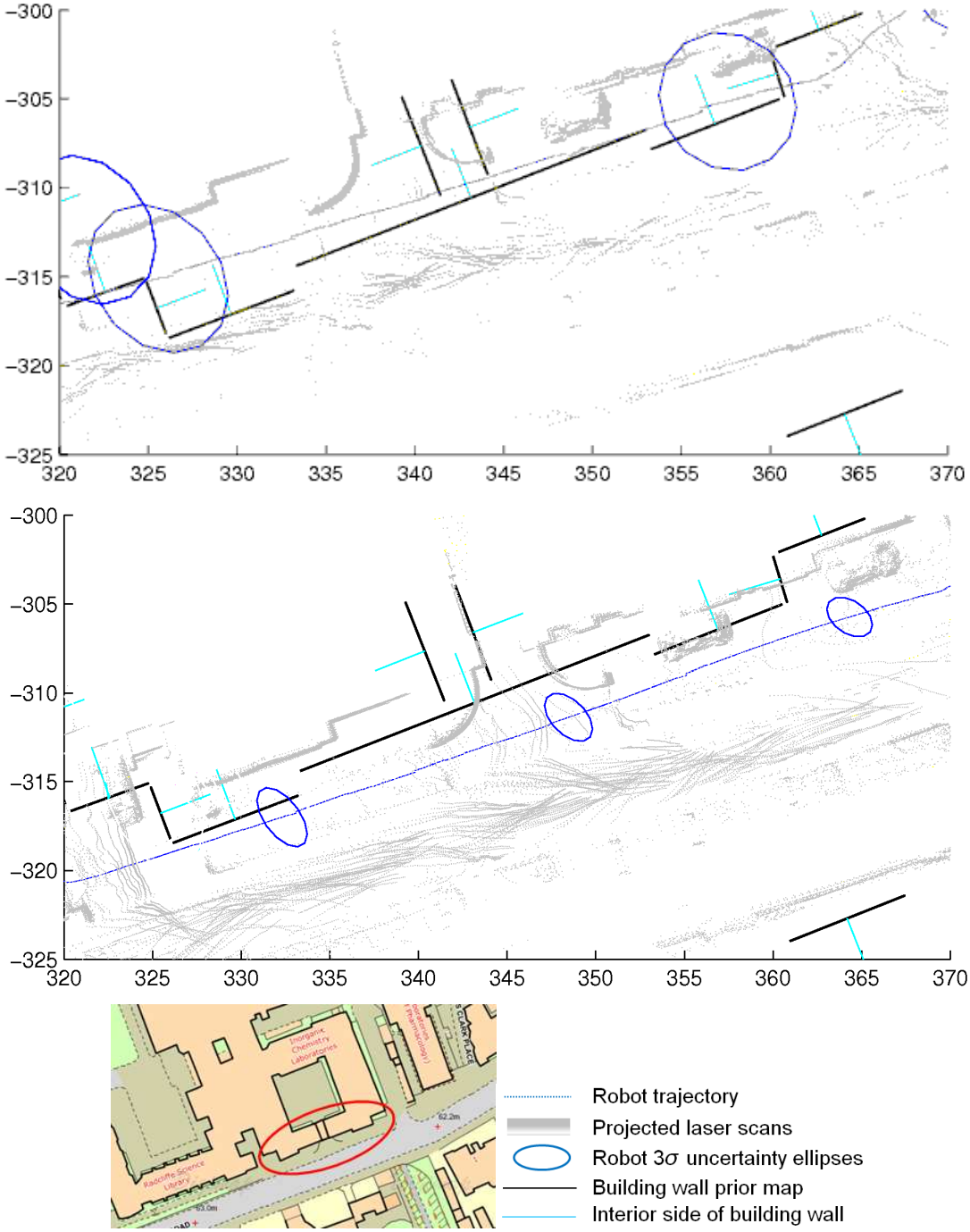

Figure 6.22: Top: Without a prior map. Bottom: With a prior map. Note how the robot trajectory appears to enter the building due to the error build up, whereas with a prior map the trajectory remains in front of the building. The blue ellipses represent the 3σ robot position uncertainty at regular intervals on the trajectory.

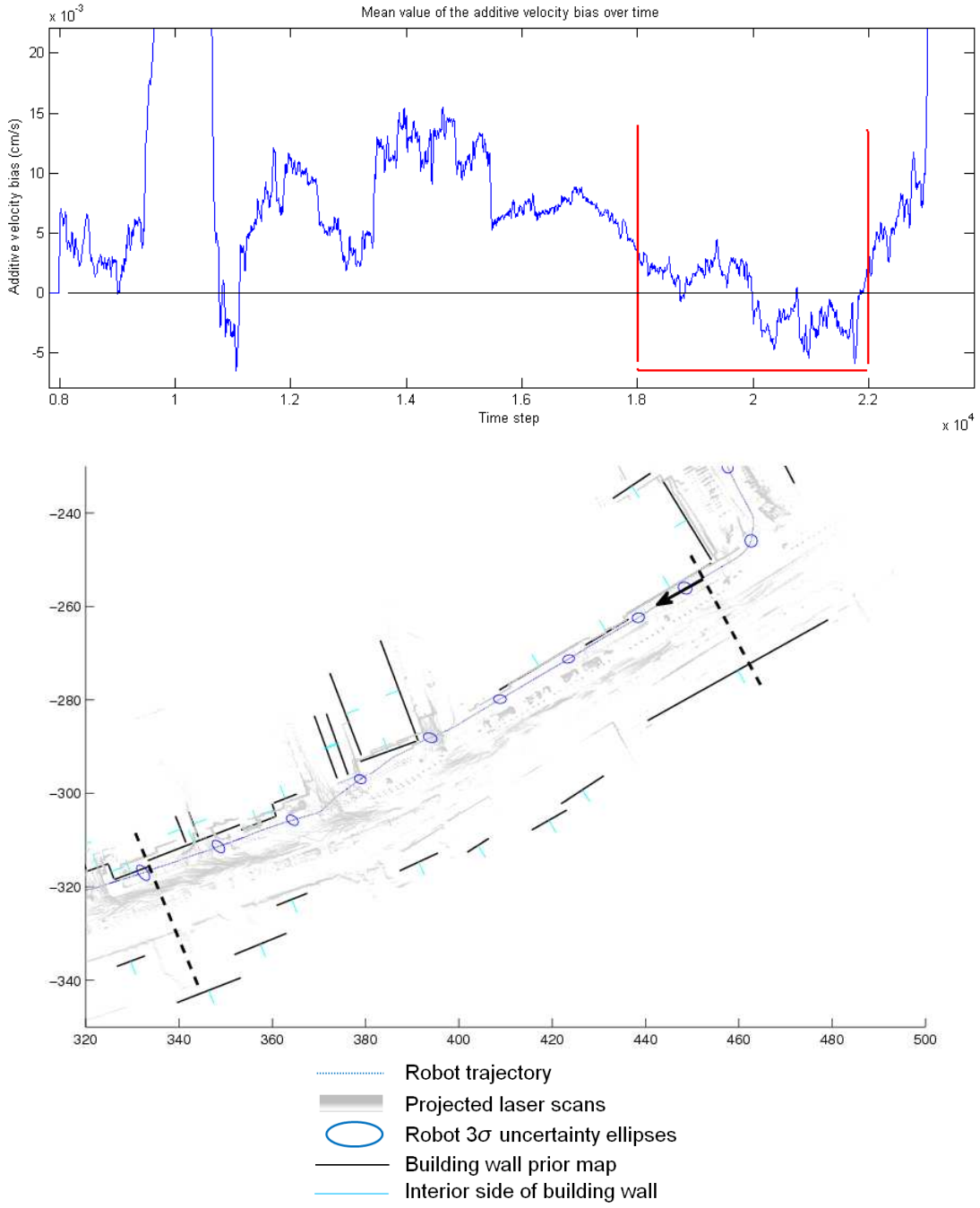

Figure 6.23: Sliding window average value (over 1000 time steps) of the velocity bias parameter $u_{\delta v}$ estimate with time step over the run, the marked region corresponding to the bottom part of the trajectory between the dashed lines (units in m). The robot travelled in the direction of the arrow, and took around 5 minutes to traverse this part of the trajectory.

Chapter 7

Summary and Future Work

7.1 Summary

In this thesis we investigated the performance of SLAM when the assumption of no prior information being available about the environment is relaxed. Although this assumption is commonly made, for most environments, such as cities a priori maps and satellite images often exist. The nature of SLAM, whereby any errors introduced propagate indefinitely limits the progress that can be achieved when restricted to the "pure" SLAM problem (that of no prior information), and as a result no single SLAM method meets all the criteria for a robust, practical SLAM system.

Fundamentally we sought to answer the question "does prior information significantly improve the SLAM problem?", concentrating on absolute prior information in the form of a sparse prior map, as this is a compelling and obvious strategy.

We focused on incremental EKF-SLAM, as it is popular and well studied. However the EKF suffers from several problems that reduce its applicability to real environments. In particular its computational (storage and update) cost, and susceptibility to linearisation and angular representation errors. We hypothesised that a geometric map can be used to mitigate these problems when integrated into EKF-SLAM.

In order to use such a prior map, we need a formal way to consider the relations between features in the SLAM state and those in the prior map, and evaluate whether these relations hold. With this in mind we formulated a unified probabilistic framework that considers the world as consisting of an underlying structure hierarchy, from which features in maps (including the SLAM state) are generated. Common underlying structure between features in two maps allow us to formulate constraints between them and thus improve their estimates.

We investigated whether prior information improves SLAM, beginning with the case where the relations between features in the SLAM state and prior map are known. We found that integrating prior information is non-trivial in an EKF-SLAM framework. While a prior map can be treated as an observation and/or parameterisation that is to be estimated jointly with the map and platform pose, the ability of the map to mitigate EKF errors is reduced, as the prior information is vulnerable to being corrupted along with the other information in the state. To prevent this we developed a novel method called the Dual Representation for using information from such a prior map in a way that is robust to

EKF errors. Using the Second Order filter with the Dual Representation, we found that the prior map reduces the platform pose error and uncertainty, even for a relatively inaccurate prior map (1 m standard deviation), and reduces the severity of linearisation and EKF errors. The improvement is significant even though the prior map consists of few features.

We then considered the realistic case where the underlying structure between features in the SLAM state and prior map must be inferred. This is a vital component of being able to practically and robustly use a prior map when performing SLAM in real world environments.

Initially we took an incremental multiple-hypothesis (MHSLAM) approach to evaluating the posterior probability, as it would allow us to apply the prior information immediately, thus reducing uncertainty and EKF errors. This involves creating a copy of the state for every likely relation hypothesis associated with an observed feature, then pruning these hypotheses based on subsequent observations. However MHSLAM results in an exponential increase in the storage and update cost with the number of hypotheses, and this makes it intractable as a practical method. To address this we developed a novel method called the Common State Filter (CSF), a generic method for resolving ambiguity in SLAM type problems. Its computational cost is still exponential, however multiple hypotheses only need to be created for part of the state. The CSF will in the worst case have the same costs as full MHSLAM, however we have found that it is generally much lower.

The CSF gave almost identical localisation performance to full MHSLAM, with half the computational update cost compared to full MHSLAM. However, the reduction is not enough to make the CSF suitable on its own for resolving relations, as the number of hypotheses still grow exponentially.

As the MHSLAM strategy is potentially too expensive computationally for general use, we decided to delay resolving and applying relations, in effect batching the observations until features in the state are sufficiently well known that there is only a single probable hypothesis. This has the advantage of only requiring a single state, and allows for the use of classifiers for detecting potential structures in each map. However it has the disadvantage that linearisation errors and EKF errors are not mitigated in the time it takes to determine whether a set of relations holds. In addition, constraints are only applied when a single hypothesis is probable, which can result in no information being used in high-clutter or sparse feature density situations.

Both MHSLAM and the single state approach were approximately equally effective at determining which relations hold. However the localisation accuracy of CSF-based MHSLAM was greater, whereas the computational performance of the single state strategy was greater.

Because of its computational tractability, we investigated the effect of the single state method when some of the line segment features in the prior map are spurious, and when varying the degree of prior map uncertainty and clutter in the environment. In general increasing clutter results in nonlinearly worse localisation performance, whereas increasing prior map error is more complicated. A very accurate prior map does not necessarily result in better performance compared to a less accurate map, because of the effect of EKF errors on the process of inferring whether relations hold. However a highly inaccurate prior map will generally give worse performance than a moderately accurate map. Even when 11% of

the line segments are spurious, and the map is relatively inaccurate (1 m standard deviation compared to the 0.1 m range error for the sensor) with a high degree of clutter (40%) the method is able to effectively discount the relation hypotheses associated with the spurious line segments.

Thus we found that in practically all of our scenarios a prior map can significantly improve the accuracy and consistency of SLAM compared to regular SLAM, even if there is high clutter and uncertainty.

7.2 Future work

There are a number of avenues of research that follow on naturally from the work in this thesis.

In chapter 5 we showed how a prior could be formulated for determining whether a relation holds between a feature in the prior map and the SLAM state. However we applied a constant as the structure relation prior in our simulations, rather than condition it on any feature observations. In future work object recognition methods such as a classifier could be implemented to improve the prior. For dense laser clouds objects can be recognised using methods such as [46], and for SLAM with a camera images and texture patches could be used, such as [53].

In this thesis we restricted ourselves to static environments for simplicity. However, the real world features many dynamic objects, from cars to swaying trees. A functional SLAM algorithm should be able to handle such dynamic objects. Two strategies for considering objects identified as being dynamic are to eliminate them from the state, or include and model them in the SLAM state. Prior information can help identify whether an object is dynamic or likely to move in the short term. For instance features detected on a road (from a prior map of roads) are likely to be cars, and thus dynamic.

In this thesis our geometric prior maps only consisted of line segments. We found these to be useful, however the nonlinearity in this representation caused problems for EKF-SLAM, and necessitated the use of the Second Order Filter. Other nonlinear parameterisations could be investigated in further work, in particular the utility of the Dual Representation in their case. In addition qualitatively different maps, such as topological and hand-drawn maps could be used. Utilising and quantifying the errors in hand-drawn maps is a major research topic, however the relative ease with which such maps can be produced, and the diversity of information they can represent makes them attractive as sources of prior information for SLAM.

Currently we assume that the prior map remains fixed for the duration of the SLAM process; that is the platform has no additional information other than from its sensors. However in reality information about the prior map could become available while the platform is performing SLAM. This could be because the prior map came from another platform performing SLAM, and is thus being constantly refined. In further work a method by which the prior map could be updated while it is being used in SLAM could be investigated. The Dual Representation in particular complicates this because the prior map is fixed.

A related problem is that of sharing maps between multiple platforms. The map being produced by a SLAM system using a prior map may itself be used by another system as a prior map. This may in

turn be shared with the original platform. Multiple platforms may also be performing cooperative (multi-agent) SLAM. This problem is non-trivial, and includes elements of data fusion, which is a research topic in itself. In an EKF framework, this is complicated by the need to share maps in a way that explicitly considers robustness to EKF errors.

Another problem is that of dealing with large correlated prior maps. The covariance matrices of such maps would be non-sparse, and would imply estimating the entire prior map along with the beacons and platform pose in the SLAM state, regardless of how many prior map features were being observed. We avoided this problem in this thesis by having a diagonal prior map covariance. In this case all the line segments in the prior map are independent, and thus we need only consider the line segments we are interested in. A possible strategy to address this is to use postponement [66]. This is an optimal method that allows EKF updates of part of the state to be postponed for any period of time, then applied in a single step. There is overhead associated with the postponement, however it is far lower than that of updating the entire state at every update. The updates associated with the majority of the prior map features could thus be postponed until those features are needed.

In this thesis we did not consider reversible data association. Thus we had no means of rescinding constraints that were applied but turned out to be unlikely based on subsequent observations. In future work reversible data association methods, such as lazy data association [51] could be applied.

We also did not consider the global localisation problem, where the initial platform starting position in the environment is completely unknown or has a very high uncertainty (for instance in the order of hundreds of metres). The work in this research could be extended to work with such high errors.

For our implementation, we restricted ourselves to EKF-SLAM, however the inability of the EKF to correctly handle nonlinearity turned out to be a significant problem. Other methods such as Fast-SLAM may provide a better framework on which to perform SLAM with prior information, for instance FastSLAM having the ability to inherently handle nonlinear process models and multiple hypotheses.

We also assumed that the SLAM system has no control over the platform's motion (passive control). However, when the SLAM system is able to control its motion, such as when used in active exploration, prior information could be useful. For instance a SLAM platform could use a prior map of a building to determine its size and tailor its search strategy appropriately, or incorporate an area for which it has a prior map. The role of prior information for making decisions such as these can be investigated.

Chapter 8

Conclusion

The aim of the research in this thesis was to investigate whether prior information, in particular in the form of a geometric prior map can improve the SLAM problem. SLAM has a number of advantages for localisation and map building over alternatives such as GPS and odometry, but suffers from a number of problems that make it difficult to solve, and as a result has found limited use in practical applications. However the "classical" SLAM approach assumes that no prior information about the environment is known. In many environments such as cities this is not true, and diverse range of prior maps are available. In this thesis we hypothesised that such information, in particular a sparse geometric prior map applied during the SLAM process can improve its performance, in particular its accuracy and consistency.

We formulated a general framework for how features in a prior map can inform those in the SLAM state, based on the concept of shared underlying structure producing features in both maps. Features with common structure have mathematical relations of a known form between them. Implementing the framework in a novel parameterisation of EKF-SLAM called the Dual Representation, we showed that for the simplified case where the structure and thus relations between the features is known, even a sparse prior map that is relatively inaccurate compared to the sensor used for SLAM significantly improves the consistency and accuracy of SLAM.

We then considered the more realistic problem where it is unknown a priori which features have common structure and thus whether relations can be applied to constrain their estimates. We showed how this can be inferred during SLAM based on the map estimates, and implemented these for EKF-SLAM based on two strategies; an incremental MHT-based approach that applies relations immediately, then tracks their hypotheses over time, and a single state method that delays applying relations until they can be resolved. The MHT approach involved developing a novel method for efficiently tracking multiple hypotheses called the Common State Filter (CSF). We found that both methods had a similar ability to resolve correct relation hypotheses, however while the CSF-based method was still computationally expensive, its undelayed application of relations produced a more accurate estimate.

We investigated the effect of spurious (non-existent) features in the prior map, and varying the degree of "clutter" (features in both maps that do not have shared structure) and accuracy of the prior map. The two are related, as the ability to use the prior map information depends on the ability to resolve the relations; as the degree of clutter or map uncertainty increases, the ability to resolve relations

decreases. However we found that even an inaccurate prior map with a high clutter level and some spurious features can still significantly improve the accuracy and consistency of EKF-SLAM.

Appendix A

Appendix

A.1 Equivalence of the Batch and Incremental Likelihoods

We can show that (4.10) is equivalent to (4.12) by showing that the likelihood terms are equivalent,

$$p(\mathbf{z}_{p\{i\}}(k)|\mathbf{z}_m, d_{in,s}=1) \propto p(\mathbf{f}(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})=0|d_{in,s}=1, \mathbf{z}_{p\{i\}}(k), \mathbf{z}_m). \tag{A.1}$$

We do this as follows. From (3.50) and (4.2),

$$\begin{aligned} p(\mathbf{x}_v, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}|d_{in,s}=1, \mathbf{z}_m, \mathbf{z}_{p\{i\}}(k)) = \\ \eta_1\, p(\mathbf{z}_{p\{i\}}(k)|\mathbf{x}_v, \mathbf{x}_{p\{i\}})\, p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}|\mathbf{z}_m, d_{in,s}=1)\, p(\mathbf{x}_v) = \\ \eta_2\, p(\theta_{in}=\phi|d_{in,s}=1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})\, p(\mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(k))\, p(\mathbf{x}_{m\{n\}}|\mathbf{z}_m)\, p(\mathbf{x}_v). \end{aligned} \tag{A.2}$$

Then for each of the two terms on the right hand side, marginalising out $\mathbf{x}_v$, $\mathbf{x}_{p\{i\}}$ and $\mathbf{x}_{m\{n\}}$ gives

$$\begin{aligned} \eta_1\, p(\mathbf{z}_{p\{i\}}(k)|\mathbf{x}_v, \mathbf{x}_{p\{i\}})\, p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}|\mathbf{z}_m, d_{in,s}=1)\, p(\mathbf{x}_v) = \\ \eta_1 \int p(\mathbf{z}_{p\{i\}}(k)|\mathbf{x}_v, \mathbf{x}_{p\{i\}})\, p(\mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}|\mathbf{z}_m, d_{in,s}=1)\, p(\mathbf{x}_v)\, \mathrm{d}\,\mathbf{x} = \\ \eta_1\, p(\mathbf{z}_{p\{i\}}(k)|\mathbf{z}_m, d_{in,s}=1), \end{aligned} \tag{A.3}$$

and

$$\begin{aligned} \eta_2\, p(\theta_{in}=\phi|d_{in,s}=1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})\, p(\mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(k))\, p(\mathbf{x}_{m\{n\}}|\mathbf{z}_m)\, p(\mathbf{x}_v) = \\ \eta_2 \int p(\theta_{in}=\phi|d_{in,s}=1, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}})\, p(\mathbf{x}_{p\{i\}}|\mathbf{z}_{p\{i\}}(k))\, p(\mathbf{x}_{m\{n\}}|\mathbf{z}_m)\, p(\mathbf{x}_v)\, \mathrm{d}\,\mathbf{x} = \\ \eta_2\, p(\theta_{in}=\phi|d_{in,s}=1, \mathbf{z}_{p\{i\}}(k), \mathbf{z}_m), \end{aligned} \tag{A.4}$$

where η represents normalising terms and $\mathbf{x} = \{\mathbf{x}_v, \mathbf{x}_{p\{i\}}, \mathbf{x}_{m\{n\}}\}$ for brevity. Thus

$$\eta_1\, p(\mathbf{z}_{p\{i\}}(k)|\mathbf{z}_m, d_{in,s}=1) = \eta_2\, p(\theta_{in}=\phi|d_{in,s}=1, \mathbf{z}_{p\{i\}}(k), \mathbf{z}_m), \tag{A.5}$$

so the two likelihoods are proportional to each other.

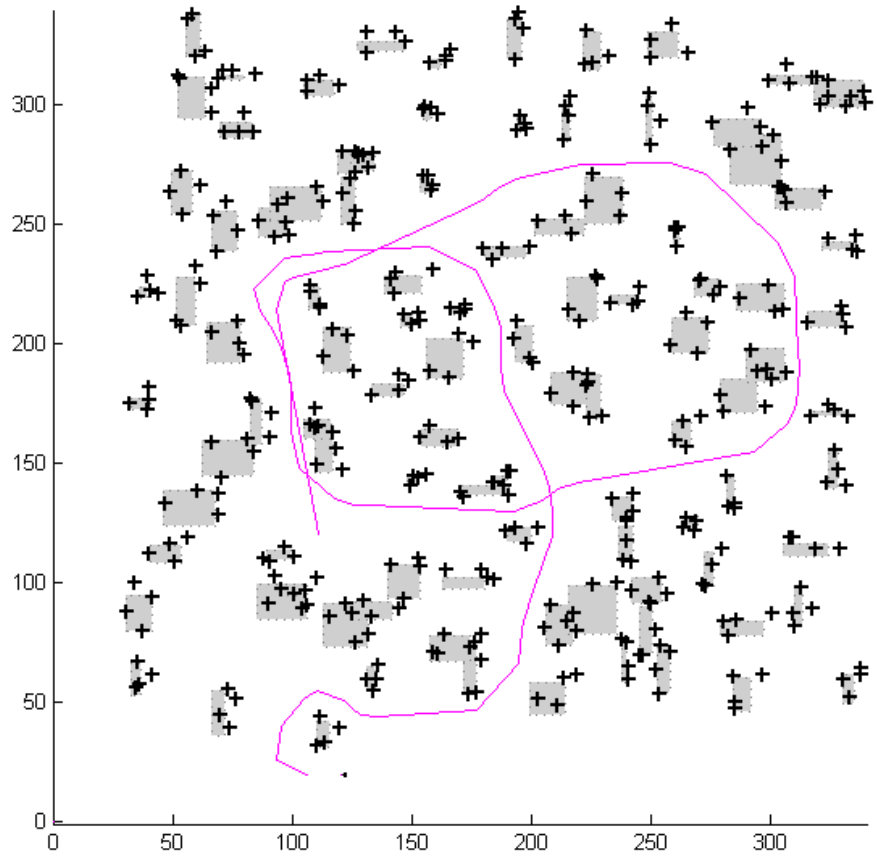

Figure A.1: The map and trajectory used in the simulations. The crosses represent beacons and the rectangles represent buildings (for occlusion). The solid line represents the vehicle trajectory, which starts at the bottom. There are two loop closures; at the top and in the centre.

A.2 Resolving Ambiguous Data Association using the CSF

In this section we apply the CSF to the well-known data association problem, that of determining which observations go with which features in the state. Ambiguities in data association can arise whenever beacons are not uniquely identifiable. The most well-known cause of this is loop-closing [15], but they can occur whenever the environment has a large number of repetitive beacons. Incorrect association can, at best, cause the creation of additional spurious beacons and, at worst, cause catastrophic filter failure. Basic approaches such as individual compatibility nearest neighbour (ICNN) are computationally simple, but most prone to data association failures [88]. Some methods for data association, such as JCBB [88], attempt to make the most informed decision using all the observations, however must do so at the time the observations are received. Thus their decisions may still be incorrect. An MHT-based strategy can be used to maintain multiple hypotheses over data association decisions. Here we implement such a strategy for EKF-SLAM, comparing the performance of full MHSLAM with the CSF in a simulated environment.

A.2.1 Simulation Studies

Simulation studies allow us to explore the properties of the algorithms in a controlled setting. We compared full MHSLAM to the CSF, with the metrics from section 4.4 in a simulated environment. The runs were performed on the map and trajectory shown in Fig. A.1. This map represents a large urban environment, with data association challenges due to the close proximity of many beacons. The distance covered is 1 km, and incorporates two loop closures. The vehicle is modelled as a steered bicycle [23] with a wheelbase of 2 m, and travels at a constant speed of $10\,ms^{-1}$, in steps of 0.02 s. The motion standard deviations were 0.1 m for the velocity and 1^o for the orientation.

The range-bearing sensor returns readings every 0.04 s (25 Hz), and has a range of 40 m and a sweep of $\pm 90^\circ$, with 0.1 m and 0.02° being the respective range and bearing noise standard deviations. The chosen motion and sensor covariances allow the vehicle to finish the trajectory in a reasonable time without becoming inconsistent or spawning excess hypotheses.

To analyse the storage and accuracy trade off, we introduce a figure of merit, representing the tradeoff between the accuracy of the vehicle and the storage requirements of a filter,

$$E(k|k) = \sqrt{\prod \text{diag}\left(\left(\mathbf{x}_v - \hat{\mathbf{x}}_v\right)\left(\mathbf{x}_v - \hat{\mathbf{x}}_v\right)^T\right)_{k|k}}\, S(k|k), \tag{A.6}$$

where diag$(\cdot)$ is the matrix diagonal, and S is the number of elements required to store the entire state covariance across all hypotheses $1 \ldots m$, which is

$$S(k|k) = \sum_{h=1}^{m} \dim(\hat{\mathbf{x}}^h(k|k))^2 \tag{A.7}$$

in the case of full MHSLAM and

$$S(k|k) = \left\{ \dim(\hat{\mathbf{x}})^2 + \sum_{h=1}^{m} \left(\dim(\hat{\mathbf{x}}^h)^2 + 2\dim(\hat{\mathbf{x}})\dim(\hat{\mathbf{x}}^h) \right) \right\}_{k|k} \tag{A.8}$$

for the CSF, where $\dim(\cdot)$ is the dimension.

The square root term in (A.6) gives us a qualitative indication of the root mean squared error (rMSE) of the x, y and θ components of the vehicle pose. This metric is more qualitative and comparative, as opposed to quantitative and absolute, partly as there is an implicit assumption as to the relative value of accuracy vs. storage (i.e. halving S is considered as valuable as a doubling of the accuracy term). However, it gives us an idea of how full MHSLAM and the variants of the CSF compare with each other. In particular, how effective the various metrics are compared to each other, and how effectively the trade off between storage/computation and accuracy (discussed in section 4.4) works in each case.

Because of the the expense associated with creating hypotheses, we took additional measures to make our implementation of MHSLAM more effective. The next section describes how we remove hypotheses that we have difficulty triming when new observations require their resources.

A.2.2 Hypothesis Management

To effectively manage the number of hypotheses and prevent memory being exhausted, the threshold value (w_t) could be increased. However, we found that simply setting values often resulted in the correct

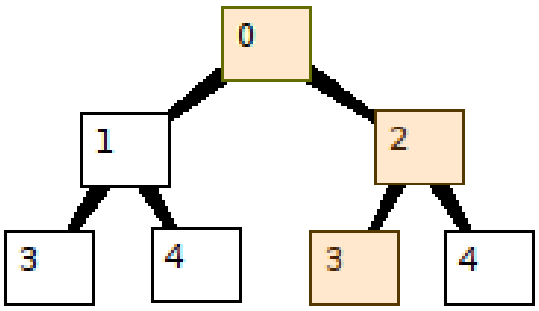

Figure A.2: Tree representation of the hypothesis states. The numbers represent association hypotheses. To remove the beacon represented by hypotheses 1 and 2, we keep the branch that contains the highest weighted hypothesis, and remove all other branches. If the highest weighted hypothesis is 0-2-3 (shaded), we would remove node 1 and its children. In our case we then delete beacon 2 but keep its children.

hypothesis being trimmed. Instead, we employed a heuristic to remove amibiguous beacons and their hypotheses. These are beacons whose observation resulted in the creation of multiple hypotheses, that could not be disambiguated in a reasonable time period. The heuristic works as follows.

If there is an ambiguous observation that we would like to use but cannot as there are too many hypotheses, we sacrifice an ambiguous beacon that has not been seen for a long time, pruning it and all of the hypotheses (filters) its observation has spawned. We keep the tree whose branch has the highest weighting and remove the others, as shown in Fig. A.2.

We perform this heuristic as many times as is required to keep the number of hypotheses manageable. This gives us improved performance as the observations of beacons close to the vehicle are prioritised over distant beacons that are unlikely to ever be trimmed, and are thus of little benefit. It should be noted that because we are only interested in determining whether two association histories are the same, maintaining the association tree is relatively cheap, with storage cost $O(mn_h)$, where m is the number of hypotheses and n_h is the number of beacons in the h hypothesis state.

The maximum number of hypotheses that can be created before memory is exhausted is h_{max}, being 250 in our case. Each hypothesis h out of a total of m is allowed to spawn off at most h_{thresh} new hypotheses, where $h_{thresh} = h_{max}/m$, thus giving all hypotheses equal potential to create new hypotheses, rather than working on a first-come first-serve basis. For a given hypothesis and observation, if the number of hypotheses spawned off by this observation will cause it to exceed h_{thresh}, the observation is discarded.

To further improve our implementation, the next section describes how we merge duplicate beacons, thus preventing persistent hypotheses being created following loop closure.

A.2.3 Merging beacons

When loop closures occur, the error in the vehicle pose can be significant enough to cause an observation of a previous beacon to be interpreted as a new beacon. Thus, following loop closure there may be a set of duplicate beacons that are actually the same physical feature. Subsequent observations of this feature create wasteful hypotheses that are difficult to resolve due to the similar beacon estimates. Thus,

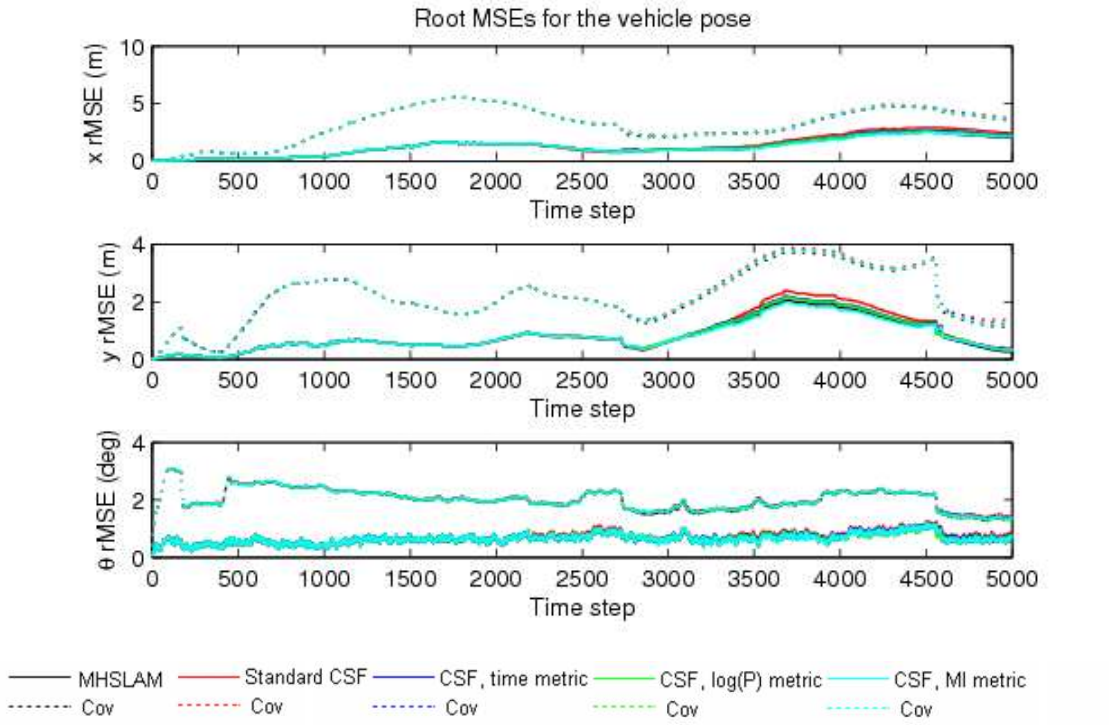

Figure A.3: Root mean squared errors for the runs, along with 2σ covariance bounds. The accuracy and covariances of the CSF variants are very similar to full MHSLAM in all instances.

to consolidate state estimates that are likely to represent the same beacon, a nearest neighbour test [88] for each beacon in the common state was performed at every time step. Beacons likely to represent the same feature had their estimates merged in a similar manner to the way we merge vehicles in section 4.3.3. We chose to merge beacons if they were closer than 1.5 m, and the exponential likelihood of their Mahalanobis distance was greater than 0.3, this arrangement being sufficient to prevent beacons from being merged incorrectly.

A.2.4 Multiple hypothesis data association

We compared full MHSLAM to the CSF and its variants when performing MHSLAM with unknown data association as discussed in section 4.2. Each simulation was averaged over 10 runs, using the same seed across all filters for a given run.

The results show that in general, the basic CSF gives similar performance to full MHSLAM, but at a vast computational and storage saving. For parts of the run the CSF even gave slightly better results than full MHSLAM.

Fig. A.3 shows that even the basic CSF gives similar rMSEs to full MHSLAM, but at a large computational and storage saving as seen from Fig. A.5. The CSF does not appear to interfere with data association or trimming, as Fig. A.5 (b) shows that the number of hypotheses throughout the run are very similar, and Fig. A.7 shows that the CSF and full MHSLAM made a similar number of incorrect associations (mainly due to observation rather than motion ambiguity).

Fig. A.4 shows that the degree of information loss in the CSF over full MHSLAM is comparable

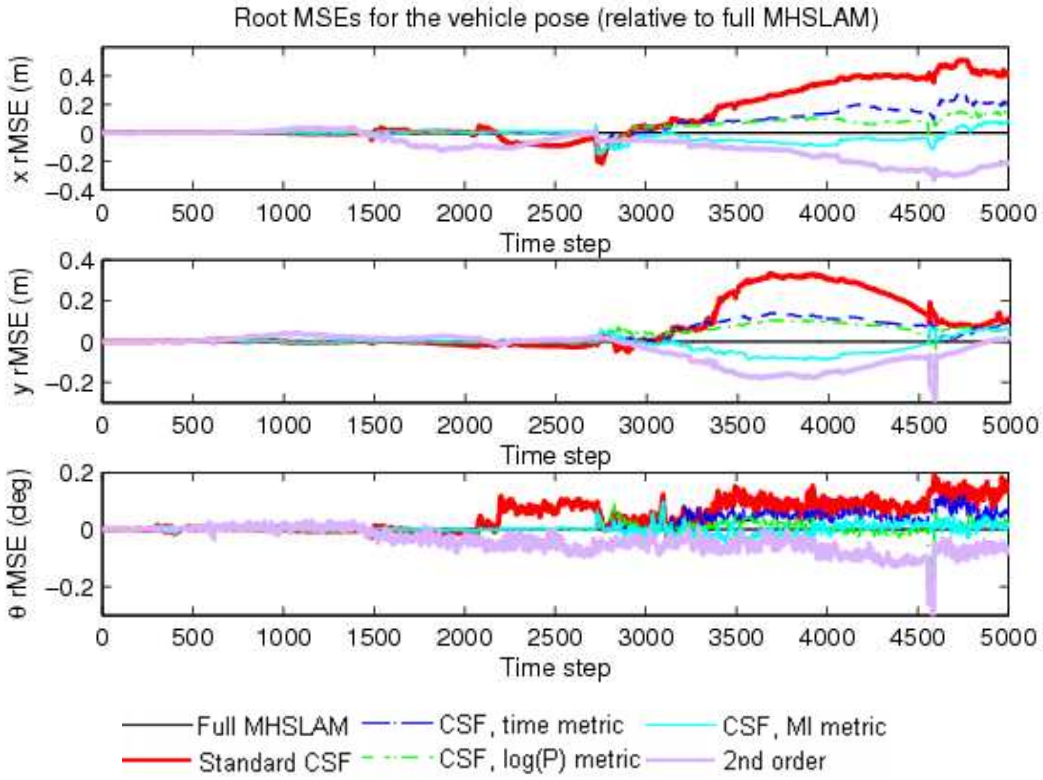

Figure A.4: Effect of different metrics on the root mean squared difference between full MHSLAM and the CSF. Although all the metrics gave some improvement, a well chosen metric gives better results can in some cases give a more accurate estimate than full MHSLAM. Full MHSLAM using second order updates has been included for comparison [56].

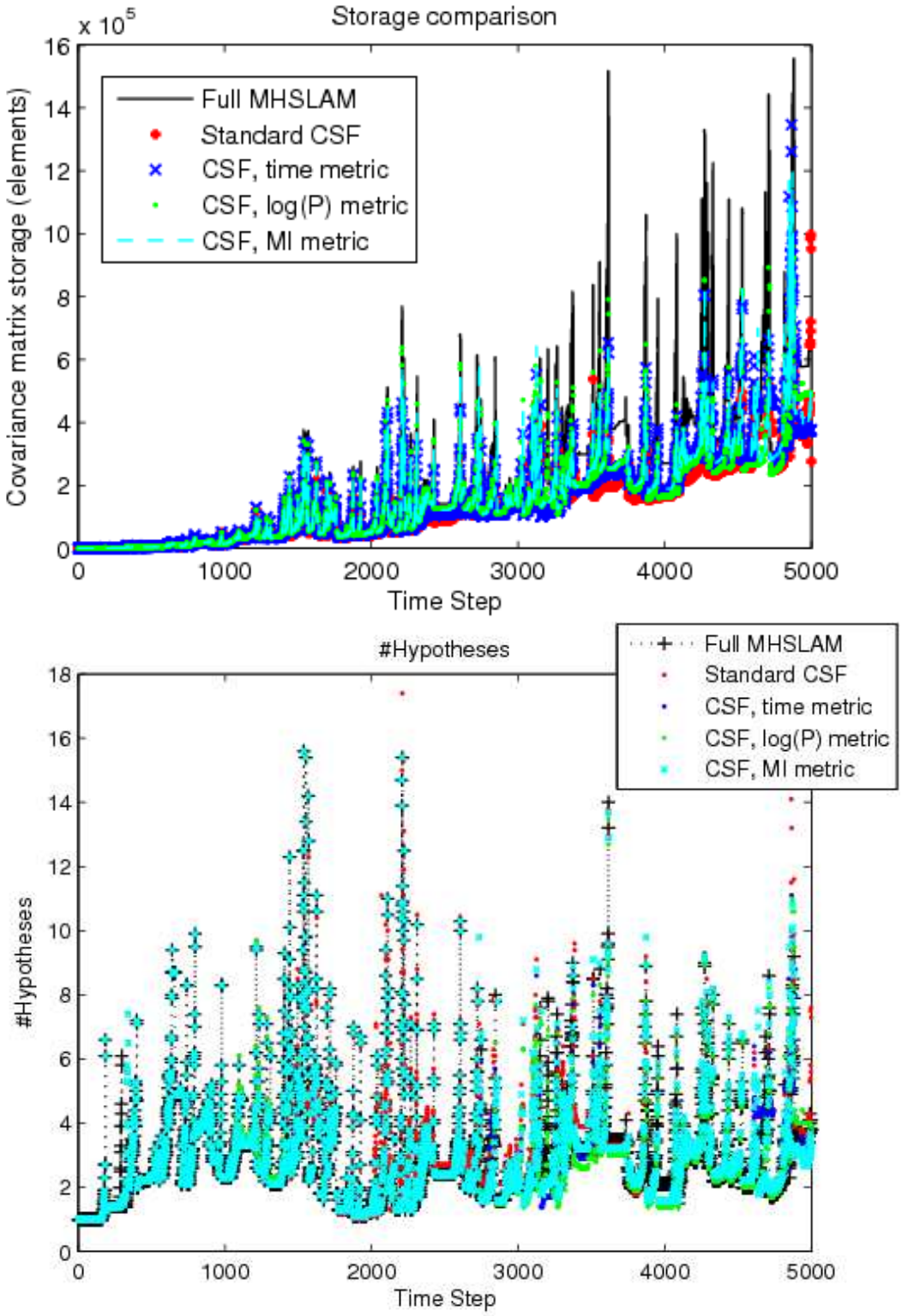

Figure A.5: (Top) Storage and (bottom) number of hypotheses between the CSF and full MHSLAM. Throughout the run, the number of hypotheses remained very similar between the two, however as the run progresses and the state size increases, the baseline and peak storage requirements increase significantly. The storage requirements of the CSF cannot exceed that of full MHSLAM. It should be noted that averaging smooths out the number of hypotheses indicated; on a per-run basis they could reach as high as 80, and thus the storage differences and efficiency metric in Fig. A.6 were more extreme in parts.

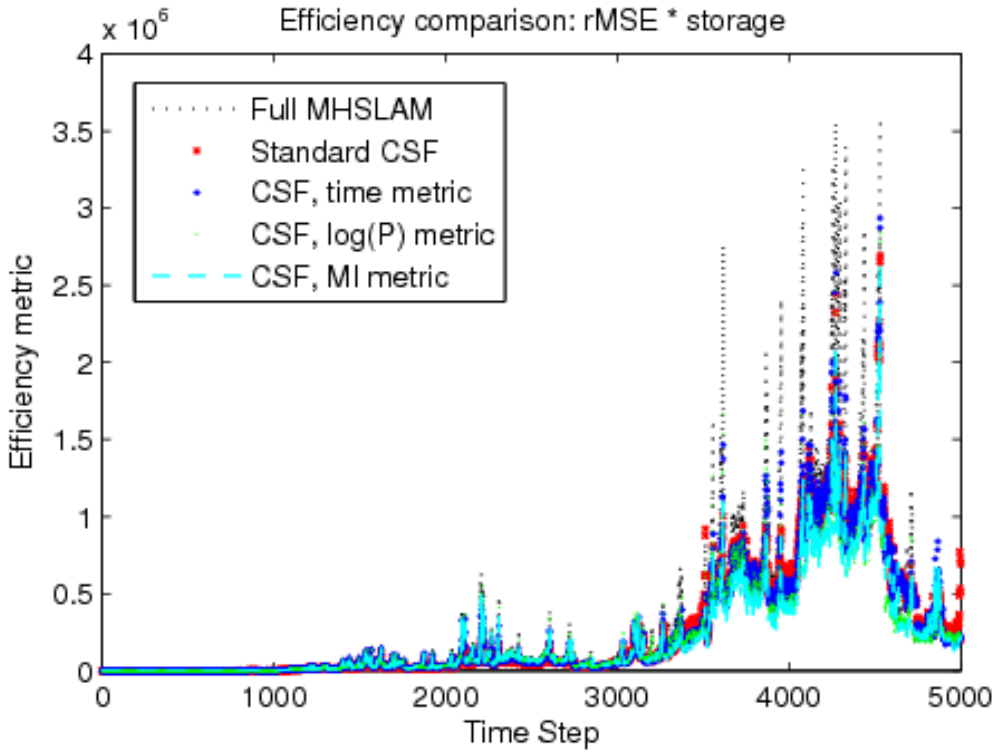

Figure A.6: The storage efficiency metric gives an idea of the storage-accuracy tradeoff in the CSF. The efficiency of all the CSF variants is similar to that of full MHSLAM, though better between 3500 and 4500 steps when there were many hypotheses. The best tested metric, based on MI, gives a slight though noticeable efficiency improvement over the other CSF variants.

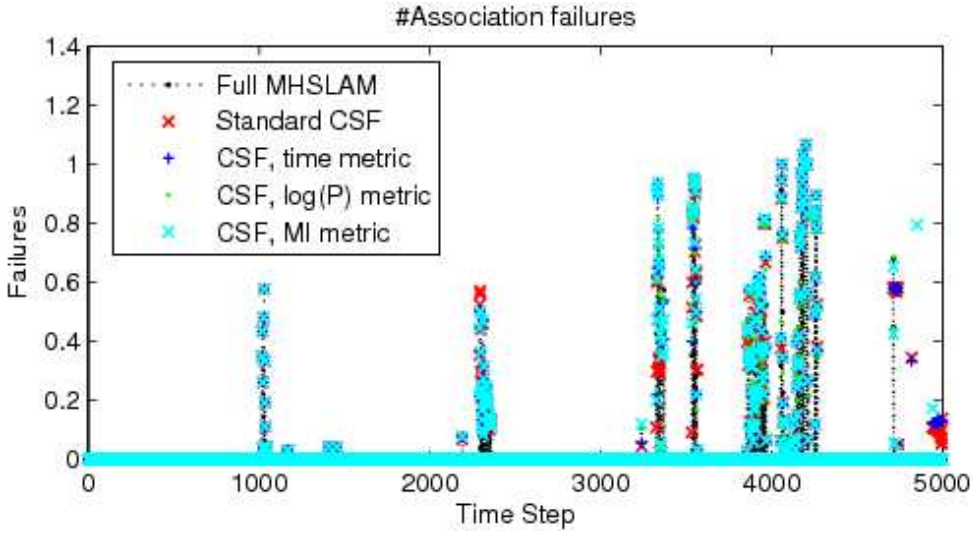

Figure A.7: The number of data association failures, that is measurements that were associated with an incorrect beacon. They remained similar between full MHSLAM and the CSF variants.

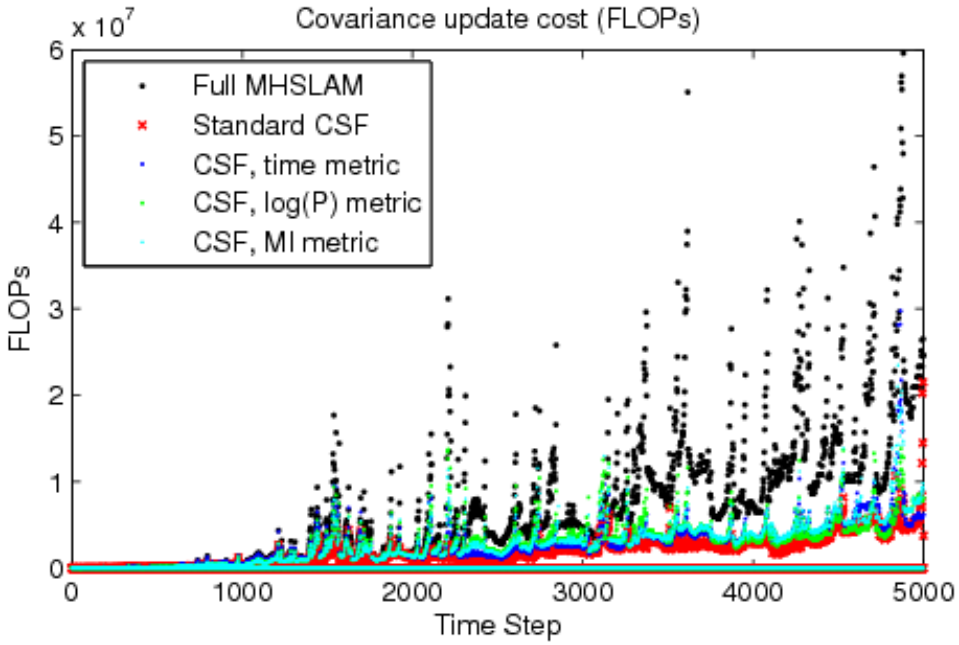

Figure A.8: The estimated number of FLOPs required to perform the Kalman update in 4.29. The computational efficiency of the CSF stands out as the run progresses and the size of the map increases. The worst-case CSF update cost cannot exceed that of full MHSLAM.

to the loss from neglecting second order terms [56] in the updates. While second order full MHSLAM gives a small improvement in accuracy, the implication is that the linearisation errors are significant, and can offset the gains from updating the entire map.

Fig. A.6 shows that the efficiency metric of the CSF variants with different metrics are very similar to full MHSLAM, except between 3500 and 4500 time steps where full MHSLAM is clearly inferior, due to the large number of hypotheses that do not contribute to a corresponding increase in accuracy. We also see that of the metrics considered, the mutual information metric gave the best tradeoff; indeed Fig. A.4 shows that it is more accurate than full MHSLAM[1] for a substantial part of the trajectory while exhibiting significant storage savings. It should be noted however that given the complexity of the problem and the large number of variables, the effect of a given metric can be unpredictable, and may not give a similar performance on other trajectories. As a result, one must be careful in the realisation of a metric, as it may result in inferior performance at a storage cost approaching that of full MHSLAM. The computational performance of the CSF variants, shown in Fig. A.8, are far superior to full MHSLAM, as they do not need to update the vast majority of the map in the common state as the number of hypotheses increases.

In general, the storage cost associated with a small increase in accuracy is very large, and can sometimes be negated by the build-up of linearisation errors. In this case the partitioned structure of the CSF can prevent linearisation errors from propagating into the map, resulting in an accuracy improvement.

[1]This is likely to be due to the Schmidt-Kalman update preventing linearisation errors from propagating through to the whole map.

A.2.5 Conclusion

The CSF can potentially give more accurate results than full MHSLAM due to the partitioned state blocking the propagation of linearisation errors. The difference in accuracy between the CSF and full MHSLAM is comparable to the difference between full MHSLAM with second order and regular Kalman updates. This provides further evidence that the linearisation errors in the Kalman filter are significant, and under certain circumstances can negate the benefit of updating the full covariance structure of EKF SLAM.

The use of a good metric, such as mutual information, to selectively transfer beacons into the hypothesis states can improve the accuracy of the CSF, whilst still maintaining a large storage and computational improvement over full MHSLAM. This also provides a mechanism by which the storage and accuracy can be traded off, depending on the platform requirements.

For futher work, the metric used to transfer beacons to the common state may be explored in more detail, and a more principled way of controlling the number of hypotheses may be investigated. Although we have only considered the augmentation of the hypothesis state, the opposite case, that of merging and transferring insignificant beacons from the hypotheses into the common state may also be considered.

A.3 Maintaining a Common State Vehicle Pose

Maintaining a common state vehicle (platform) pose is an alternative to moving observed beacons from the common state into the hypothesis state to update them. By keeping these beacons in the single common state, the storage and computational requirements of the filter are reduced compared to moving these beacons into each hypothesis state.

However, there is a degree of information loss present in this scheme, as the common state vehicle uncertainty is greater than those of the hypothesis state vehicles, on account of it not receiving information from beacons being updated in the hypothesis states. In our experiments we found that using the common state vehicle strategy gave inferior trimming performance, so we do not use it. However it may work well in other applications.

The common state vehicle is used for the purpose of updating the common and hypothesis states immediately after the merging stage in section 4.3.3. We did not do this in our experiments, as we found it to degrade the CSF performance, however it is mentioned as it may have future merit.

If used, the comon state in (4.18) gains a common state vehicle $\hat{\mathrm{x}}_v$ to become

$$\hat{\mathbf{x}}^h(k|k-1) = \left[\begin{array}{c} \left[\begin{array}{cc} \hat{\mathrm{x}}_v^T & \hat{\mathrm{x}}_b^T \end{array} \right] \\ \left[\begin{array}{cc} \hat{\mathbf{x}}_v^T & \hat{\mathbf{x}}_b^T \end{array} \right]^h \end{array} \right]^T_{k|k-1}, \tag{A.9}$$

and the covariance (4.19) of hypothesis h becomes

$$\mathbf{P}^h(k|k-1) = \left[\begin{array}{cc} \left[\begin{array}{cc} \mathrm{P}_v & \mathrm{P}_{vb}^T \\ \mathrm{P}_{vb} & \mathrm{P}_b \end{array} \right] & \left[\begin{array}{cc} \mathrm{P}_{vv}^T & \mathrm{P}_{vb}^T \\ \mathrm{P}_{bv}^T & \mathrm{P}_{bb}^T \end{array} \right]^h \\ \left[\begin{array}{cc} \mathrm{P}_v & \mathrm{P}_{vb}^T \\ \mathrm{P}_{vb} & \mathrm{P}_b \end{array} \right]^h & \left[\begin{array}{cc} \mathrm{P}_v & \mathrm{P}_{vb}^T \\ \mathrm{P}_{vb} & \mathrm{P}_b \end{array} \right]^h \end{array} \right]_{k|k-1}. \tag{A.10}$$

The form of the prediction Jacobians $\nabla G^h(k)$ and $\nabla F^h(k)$ in section 4.3.1 is

$$\nabla G^h(k) = \begin{bmatrix} \nabla G_c & \mathbf{0} & \nabla G_h^h & \mathbf{0} \end{bmatrix}_k^T, \tag{A.11}$$

and

$$\nabla F^h(k) = \begin{bmatrix} \nabla F_c(k) & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \nabla F_h^h(k) & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix}, \tag{A.12}$$

with respect to (4.18), where $\nabla F_c(k)$ is the common state vehicle Jacobian and $\nabla F_h^h(k)$ is the hypothesis state vehicle Jacobian for the hypothesis h, linearised about their respective vehicle estimates. The form of $\nabla G^h(k)$ allows the common and hypothesis vehicles to receive common process noise.

Although we do not use $\hat{\mathtt{x}}_v$ to update beacons, it is correlated with $\hat{\mathtt{x}}_v^h$ (as seen by the $\mathtt{p}_{\mathtt{vv}}$ term in A.10) through common process noise $\mathbf{v}$. Thus although we block information from passing from the hypothesis states to the common state, the correlation between the common state and hypothesis state vehicle allows us to propagate information from $\mathtt{P}_\mathtt{b}$ (the hypothesis state beacons) to $\mathtt{P}_b$ (the common state beacons) when we come to merge the hypothesis and common states.

Before transfering beacons into the common state as in section 4.3.3, we merge the vehicle poses by performing a Kalman update to constrain both vehicle states to have the same estimate. The pseudo-observation [112] is thus $\hat{z} = \hat{\mathtt{x}}_v - \hat{\mathtt{x}}_v^c$. As both vehicle states represent the same vehicle the true value is $\mathbf{z} = \mathbf{0}$. There is no noise in the pseudo-observation hence $\mathbf{R} = \mathbf{0}$. Following this step the hypothesis state beacons may now be transferred into the common state as above, and one of the vehicles discarded.

Bibliography

[1] *Maintaining representations of the environment of a mobile robot*, Cambridge, MA, USA, 1988. MIT Press.

[2] Anuraag Agrawal, Atsushi Nakazawa, and Haruo Takemura. Mmm-classification of 3d range data. In *International Conference on Robotics and Automation*, pages 2003–2008, 2009.

[3] D. Alspach and H. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *Automatic Control, IEEE Trans.*, 17(4):439–448, 1972.

[4] B.D.O. Anderson and J.B. Moore. *Optimal filtering*. Prentice-Hall Information and System Sciences Series, Englewood Cliffs: Prentice-Hall, 1979, 1979.

[5] M. Awrangjeb and Guojun Lu. Robust image corner detection based on the chord-to-point distance accumulation technique. *Multimedia, IEEE Transactions on*, 10(6):1059–1072, Oct. 2008.

[6] T Bailey and H Durrant-Whyte. Simultaneous localisation and mapping (slam) part 1: The essential algorithms. *Robotics and Automation Magazine*, 2006.

[7] T Bailey and H Durrant-Whyte. Simultaneous localisation and mapping (slam) part 2: State of the art. *Robotics and Automation Magazine*, 2006.

[8] T. Bailey, J. Nieto, and E. Nebot. Consistency of the fastslam algorithm. In *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE International Conference on*, pages 424–429, 2006.

[9] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568, 2006.

[10] Y. Bar-Shalom and T. E. Fortmann. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[11] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[12] Kristopher R. Beevers and Wesley H. Huang. Inferring and enforcing relative constraints in SLAM. In *2006 Workshop on the Algorithmic Foundations of Robotics (WAFR 2006)*, New York, NY, July 2006.

[13] K.E. Bekris, M. Glick, and L.E. Kavraki. Evaluation of algorithms for bearing-only slam. In *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE International Conference on*, pages 1937–1943, 2006.

[14] C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Proceedings of Robotics Science and Systems*, 2007.

[15] M. Bosse, P. Newman, J. Leonard, and S. Teller. An atlas framework for scalable mapping, 2002.

[16] Ronald N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill, New York :, 2d ed. edition, 1978.

[17] Mitch Bryson and Salah Sukkarieh. Building a robust implementation of bearing-only inertial slam for a uav. *J. Field Robotics*, 24(1-2):113–143, 2007.

[18] J.A. Castellanos, J.D. Tardos, and G. Schmidt. Building a global map of the environment of a mobile robot: the importance of correlations. In *Robotics and Automation, 1997. Proc., 1997 IEEE International Conference on*, volume 2, pages 1053–1059, 1997.

[19] Jose A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15:948–953, 1999.

[20] H.J. Chang, C.S.G. Lee, Y. Lu, and Y.C. Hu. Simultaneous localization and mapping with environmental structure prediction. In *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE International Conference on*, pages 4069–4074, 2006.

[21] Antonio Chella, Massimo Cossentino, Roberto Pirrone, and Andrea Ruisi. Modeling ontologies for robotic environments. In *Proc. of the fourteenth international conference on software engineering and knowledge engineering*, pages 15–19, 2002.

[22] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Trans.*, 17(2):125–137, 2001.

[23] S. Clark and H. Durrant-Whyte. Autonomous land vehicle navigation using millimeter wave radar. In *Robotics and Automation, 1998. Proc. 1998 IEEE International Conference on*, volume 4, pages 3697–3702, 1998.

[24] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International J. Robotics Research*, 27(6):647–665, 2008.

[25] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proc. Ninth IEEE International Conference on*, volume 2, pages 1403–1410, 2003.

[26] A.J. Davison. Active search for real-time vision. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 66–73, 2005.

[27] A.J. Davison, Y. González Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon*, 2004.

[28] Andrew J. Davison and Nobuyuki Kita. 3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain, 2001.

[29] Andrew J. Davison and David W. Murray. Simultaneous localization and map-building using active vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 24, pages 865–880, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[30] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[32] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Trans.*, 17(3):229–241, 2001.

[33] Yong Du, B. Guindon, and J. Cihlar. Haze detection and removal in high resolution satellite image with wavelet analysis. *Geoscience and Remote Sensing, IEEE Transactions on*, 40(1):210–217, 2002.

[34] T. Duckett. A genetic algorithm for simultaneous localization and mapping. In *Robotics and Automation, 2003. Proc. ICRA '03. IEEE International Conference on*, volume 1, pages 434–439, 2003.

[35] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.

[36] H.F. Durrant-Whyte. Uncertain geometry in robotics. *Robotics and Automation, IEEE J. [see also IEEE Transactions on Robotics and Automation]*, 4(1):23–31, 1988.

[37] A.I. Eliazar and R. Parr. Dp-slam 2.0. In *Robotics and Automation, 2004. Proc. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1314–1320, 2004.

[38] J. Farrell and M. Livstone. Exact calculations of discrete-time process noise statistics for hybrid continuous/discrete time applications. In *Decision and Control, 1993., Proc. the 32nd IEEE Conference on*, volume 1, pages 857–858, 1993.

[39] J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *Robotics and Automation, 2004. Proc. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 383–390, 2004.

[40] U. Frese. A proof for the approximate sparsity of slam information matrices. In *Robotics and Automation, 2005. ICRA 2005. Proc. the 2005 IEEE International Conference on*, pages 329–335, 2005.

[41] U. Frese and G. Hirzinger. Simultaneous localization and mapping - a discussion. In *Proc. the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle*, 2001.

[42] Udo Frese. Treemap: An o(log n) algorithm for simultaneous localization and mapping. In *In spatial cognition IV*, pages 455–476. Springer Verlag, 2005.

[43] Udo Frese. A discussion of simultaneous localization and mapping. *Auton. Robots*, 20(1):25–42, 2006.

[44] Andrew P. Gee, Denis Chekhlov, Walterio Mayol, and Andrew Calway. Discovering planes and collapsing the state space in visual slam. In *Proc. the 18th British Machine Vision Conference*, September 2007.

[45] A. Georgiev and P.K. Allen. Localization methods for a mobile robot in urban environments. *Robotics, IEEE Trans. [see also Robotics and Automation, IEEE Trans.]*, 20(5):851–864, 2004.

[46] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision (ICCV)*, Sep 2009.

[47] J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Trans.*, 17(3):242–257, 2001.

[48] Jose Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001.

[49] Jose Guivant, Eduardo Nebot, Juan Nieto, and Favio Masson. Navigation and Mapping in Large Unstructured Environments. *The International J. Robotics Research*, 23(4-5):449–472, 2004.

[50] Jose E. Guivant, Favio R. Masson, and Eduardo M. Nebot. Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 40(2-3):79–90, August 2002.

[51] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun. Towards lazy data association in SLAM. In *Proc. the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy, 2003. Springer.

[52] Joon H. Han and Timothy Poston. Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature. *Pattern Recognition Letters*, 22(10):1133–1144, 2001.

[53] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2137–2144, June 2006.

[54] D.A. Holland, D.S. Boyd, and P. Marshall. Updating topographic mapping in great britain using imagery from high-resolution satellite sensors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(3):212–223, 2006.

[55] Patrick Y. C. Hwang and Robert Grover Brown. *Introduction to Random Signals and Applied Kalman Filtering*. J. Wiley, 1992.

[56] A. H. Jazwinski. *Stochastic Processes and filtering theory*. Academic Press, 1970.

[57] S. Julier. The stability of covariance inflation methods for slam. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proc. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2749–2754, 2003.

[58] S. Julier. Consistency and slam. 2006.

[59] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[60] S. J. Julier and H. F. Durrant-Whyte. A horizontal model fusion paradigm. In *The Proceedings of the SPIE AeroSense Conference: Navigation and Control Technologies for Unmannged Systems*, volume 2738, pages 37–48, Orlando, FL, USA, 1996.

[61] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proc. 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243, 2001.

[62] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics, TRO*, 24(6):1365–1378, Dec 2008.

[63] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proc. 1999 IEEE International Conference on*, volume 6, pages 739–743, 1999.

[64] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[65] Alexander Kleiner, Johann Prediger, and Bernhard Nebel. Rfid technology-based exploration and slam for search and rescue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4054–4059, 2006.

[66] J. Knight, A. Davison, and I. Reid. Towards constant time slam using postponement. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 1:405–413, 2001.

[67] K. Konolige, M. Agrawal, and Joan Solà. Large scale visual odometry for rough terrain. In *Proc. International Symposium on Research in Robotics (ISRR)*, November 2007.

[68] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Technical Report AI90-120, 1, 1990.

[69] R. Kummerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[70] N.M. Kwok and G. Dissanayake. An efficient multiple hypothesis filter for bearing-only slam. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proc. 2004 IEEE/RSJ International Conference on*, volume 1, pages 736–741, 2004.

[71] N.M. Kwok, G. Dissanayake, and Q.P. Ha. Bearing-only slam using a sprt based gaussian sum filter. In *Robotics and Automation, 2005. Proc. the IEEE International Conference on*, pages 1109–1114, 2005.

[72] Kwang Wee Lee, Sardha Wijesoma, and Javier Ibanez Guzman. A constrained slam approach to robust and accurate localisation of autonomous ground vehicles. *Robotics and Autonomous Systems*, 55(7):527–540, 2007.

[73] T. Lemaire, S. Lacroix, and J. Sola. A practical 3d bearing-only slam algorithm. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2449–2454, 2005.

[74] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006.

[75] Xiao-Rong Li and Y. Bar-Shalom. Multiple-model estimation with variable structure. *Automatic Control, IEEE Trans.*, 41(4):478–493, Apr 1996.

[76] Benson Limketkai, Lin Liao, and Dieter Fox. Relational object maps for mobile robots. In *IJCAI*, pages 1471–1476, 2005.

[77] Yufeng Liu and S. Thrun. Results for outdoor-slam using sparse extended information filters. In *Robotics and Automation, 2003. Proc. ICRA '03. IEEE International Conference on*, volume 1, pages 1227–1233, 2003.

[78] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[79] R. Madhavan, G. Dissanayake, and H.F. Durrant Whyte. Map-building and map-based localization in an underground-mine by statistical pattern matching. volume 2, pages 1744–1746, 1998.

[80] D. Maksarov and H. Durrant-Whyte. Mobile vehicle navigation in unknown environments: a multiple hypothesis approach. *Control Theory and Applications, IEE Proc.*, 142(4):385–400, 1995.

[81] M. C. Malin, M. H. Carr, G. E. Danielson, M. E. Davies, W. K. Hartmann, A. P. Ingersoll, P. B. James, H. Masursky, A. S. McEwen, L. A. Soderblom, P. Thomas, J. Veverka, M. A. Caplinger, M. Ravine, T. A. Soulanille, and J. L. Warren. Early views of the martian surface from the mars orbiter camera of mars global surveyor. *Science*, 279(5357):1681–1685, March 1998.

[82] Rogan Mendoza and Mary-Anne Williams. Ontology based object categorisation for robots. In *AOW '05: Proceedings of the 2005 Australasian Ontology Workshop*, pages 61–67, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[83] P. M.Newman. On the solution to the simultaneous localization and map building problem. *Australian Centre for Field Robotics*, 1999.

[84] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[85] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *In Proc. of the Int. Conf. on Artificial Intelligence*, pages 1151–1156, 2003.

[86] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and Jose Santos-Victor. Affordances, development and imitation. In *IEEE - International Conference on Development and Learning*, 2007.

[87] J. Montiel, J. Civera, and Andrew Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *Robotics: Science and Systems*, August 2006.

[88] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Trans.*, 17(6):890–897, 2001.

[89] J.I. Nieto, J.E. Guivant, and E.M. Nebot. The hybrid metric maps (hymms): a novel map representation for denseslam. In *Robotics and Automation, 2004. Proc. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 391–396, 2004.

[90] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proc. of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652–I–659, 2004.

[91] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam—3d mapping outdoor environments: Research articles. *J. Field Robot.*, 24(8-9):699–722, 2007.

[92] Andreas Nüchter, Oliver Wulf, Kai Lingemann, Joachim Hertzberg, Bernado Wagner, and Hartmut Surmann. 3d mapping with semantic knowledge. In *In RoboCup International Symposium*, pages 335–346, 2005.

[93] M. A. Oliver and R. Webster. Kriging: a method of interpolation for geographical information systems. *International journal of geographical information systems*, 4(3):313–332, 1990.

[94] Ordnance Survey. *OS Mastermap Part 1: user guide*, 04 2006.

[95] M. Parsley and S. Julier. Slam with a heterogeneous prior map. In *Proceedings of the Fourth Annual SEAS DTC Conference*, 2009.

[96] Martin P. Parsley and Simon J. Julier. The common state filter for slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2060–2065, 2008.

[97] Mark A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1157–1164, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[98] Ingmar Posner, Mark Cummins, and Paul Newman. Fast probabilistic labeling of city maps. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[99] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044–1049, Oct 2007.

[100] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24:843–854, 1979.

[101] Branko Ristic and Sanjeev Arulampalam. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House radar library, 2004.

[102] D. Rodriguez-Losada, F. Matia, A. Jimenez, and R. Galan. Consistency improvement for slam - ekf for indoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE International Conference on*, pages 418–423, 2006.

[103] S. F. Schmidt. Application of state space methods to navigation problems. *Advanced Control Systems*, 3:293–340, 1966.

[104] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[105] Gabe Sibley. A sliding window filter for slam. Technical report, University of Southern California, 2006.

[106] C. Smith. *Integrating Mapping and Navigation*. PhD thesis, Massachusetts Institute of Technology, 1998.

[107] Paul Smith, Ian Reid, and Andrew Davison. Real-Time Monocular SLAM with Straight Lines. In *British Machine Vision Conference*, volume 1, pages 17–26, September 2006.

[108] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Robotics and Automation. Proc. 1987 IEEE International Conference on*, volume 4, pages 850–850, 1987.

[109] J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only slam. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2499–2504, Aug. 2005.

[110] H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657 –2664, May 2010.

[111] S. Sukkarieh A. Sanfeliu T. Vidal-Calleja, M. Bryson and J. Andrade-Cetto. On the observability of bearing only slam. *Proc. the IEEE International Conference on Robotics and Automation*, 2007.

[112] M. Tahk and J.L. Speyer. Target tracking problems subject to kinematic constraints. *Automatic Control, IEEE Trans.*, 35(3):324–326, 1990.

[113] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, G. Lakemeyer, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, and W. Steiner. Experiences with two deployed interactive tour-guide robots. In *Proceedings of the International Conference on Field and Service Robotics*, Pittsburgh, PA, 1999.

[114] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[115] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[116] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. accepted for publication.

[117] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association, 2004.

[118] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[119] Sebastian Thrun. *Robotic mapping: a survey*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[120] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1-3):29–53, 1998.

[121] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International J. Robotics Research*, 23(7-8):693–716, 2004.

[122] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *Int. J. Rob. Res.*, 25(5-6):403–429, 2006.

[123] Matthew R. Walter, Ryan M. Eustice, and John J. Leonard. Exactly sparse extended information filters for feature-based slam. *The International Journal of Robotics Research*, 26(4):335–359, 2007.

[124] E.A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[125] J. Weingarten and R. Siegwart. Ekf-based 3d slam for structured environment reconstruction. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3834–3839, 2005.

[126] Stefan B. Williams, Hugh Durrant-Whyte, and Gamini Dissanayake. Constrained Initialization of the Simultaneous Localization and Mapping Algorithm. *The International J. Robotics Research*, 22(7-8):541–564, 2003.

[127] B. Yamauchi, A. Schultz, and W. Adams. Mobile robot exploration and map-building with continuous localization. In *Robotics and Automation, 1998. Proc. 1998 IEEE International Conference on*, volume 4, pages 3715–3720, 1998.

[128] Jooseop Yun and J. Miura. Multi-hypothesis outdoor localization using multiple visual features with a rough map. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3526–3532, 2007.